AllFusion™ Harvest Change Manager

Getting Started

5.1.1

Computer Associates™

MAN04141804E

Contents

Chapter 1

Developers, Ready To Get Synchronized?

Unsurpassed Change And Configuration Management

Today's development teams build large, distributed application systems. They work from heterogeneous platforms at remote locations and make simultaneous changes to a multitude of interrelated software modules and system documentation. The only way to effectively track this complex, enterprise-wide development activity is with a comprehensive, repository-based, web-enabled change and configuration management (CCM) solution with an open architecture. Manual methods and simple version control systems just aren't robust enough to help you improve the development process and bolster service levels.

AllFusion Harvest Change Manager (also referred to simply as Harvest) helps you synchronize development activities on multiple platforms, across your enterprise and throughout the application development life cycle. It also scales up to serve project teams working on your largest client/server enterprise systems and scales down to meet the needs of individual developers. Through bi-directional integration with AllFusion Endevor Change Manager (also referred to simply as Endevor), Harvest extends CCM support to the well-managed mainframe, providing unparalleled levels of CCM support for the enterprise.

The Purpose Of This Guide

This book introduces AllFusion Harvest Change Manager to you in an efficient and visual manner. By the time you have finished reading this guide, you will have an overview of the wide scope of the product and its usability will be familiar to you. It is important to us that you feel comfortable with AllFusion Harvest Change Manager before you begin to use it.

A Centralized Vantage Point

Production-oriented development environments focus on keeping existing applications operational. Typically, there's a constant flow of "small" changes to the production version—changes that are incorporated as they are completed, so that new features and information can be made available to users as soon as possible.

Harvest provides a centralized management point for streamlining and coordinating software change processes throughout your distributed environment by tracking and "packaging" application components.

By automating the introduction of changes and then streamlining their migration into production, Harvest protects your production application from contamination due to undesired change. As a result, Harvest eliminates development crises and ensures a smooth transition between development life cycle phases.

Construct Your Own Development Process

Traditional CCM tools make assumptions about how your organization works. They require programming resources for any modification. With Harvest, it's business as usual! Harvest helps you create and modify models of your own development processes through simple point-and-click and drag-and-drop operations. It then uses your model to keep software changes under control, schedules on track, and everyone up-to-date. By automating the workflow, many routine application development tasks are also automated, including notifications, approvals and change migrations from one phase to another.

Harvest includes a set of packaged life cycle models with predefined phases and processes that map to common development environments. This enables you to select an easily customized model to represent your environment, significantly reducing the time and effort usually required for implementing a CCM solution.

Notification Eliminates Confusion

A simple Harvest set-up procedure allows you to select concurrent project development. Concurrent development prevents one developer from overwriting another developer's changes. Harvest automatically isolates changes into separate sets. Participating developers can be automatically notified whenever concurrent development occurs; reports that show what changed, why it changed, who made the changes and when it was changed, are easily generated!

You can combine the changes of all concurrent activities with the integrated merge facility, viewing and resolving any conflicts among versions as necessary.

Manage Parallel Development Activities

When you have to maintain multiple releases of the same application, Harvest provides you with the tools you need. The integrated merge facility enables you to automate the merging of some or all of those changes into a subsequent development process, eliminating labor-intensive manual merges. Both short and long-term projects can be developed in parallel, without inadvertently affecting one another.

Navigate Efficiently

The Harvest Workbench facilitates faster, more efficient navigation through your CCM environment. It combines ease-of-use and a familiar look and feel with the power to execute day-to-day configuration operations. The Harvest Workbench benefits both new and advanced Harvest users because it does not require a great deal of training to navigate through the product's features and functionality.

Synchronize Enterprise Development Activities

For unparalleled levels of enterprise CCM support, Harvest integrates with Endevor through AllFusion Change Manager Enterprise Workbench, which provides a central management point for all package migrations. AllFusion Change Manager Enterprise Workbench ensures that application changes in both Harvest and Endevor can be packaged, approved, and moved into production in both environments together, and also ensures that projects only be deployed when all platforms are ready for production.

Open Architecture

Harvest was designed with a very open architecture. This open architecture allows users to take advantage of most CCM functions external from the user interface itself. Through the use of this open architecture, users have begun to use Harvest as the focal point for integrating application development with traditionally systems-oriented functions, such as automation and helpdesk tools.

Through this approach, Harvest enables users to provide end-to-end control.

Harvest's open architecture allows easy access to CCM information. Rather than developing yet another database standard, control information is stored within a commercially available relational database—available to any user!

Harvest's Distinctive Features

- Readily adapts to your development process
- Enables synchronization of concurrent development activities
- Allows you to develop multiple projects in parallel, and then merge them together as needed
- Allows you to track information associated with every change
- Facilitates integration with other development and maintenance tools through an open architecture and command line interface
- Employs an open architecture to enable easy access to CCM information
- Provides a complete solution that encompasses all facets of the software development and maintenance process
- Provides a consistent interface across all development and maintenance platforms in the environment
- Allows you to navigate efficiently with the Harvest Workbench and web interface.
- Easy to use

CA Services: Enabling Solutions Through Experience

When it comes to getting on the information fast track, CA Services can recommend and install a full suite of portal and knowledge management solutions to keep your business moving. And our associates offer the proprietary know-how on custom-fitting your enterprise for solutions ranging from life cycle management, data warehousing, and next-level business intelligence. Our experts will leave you with the technology and knowledge tools to fully collect, exploit, and leverage your data resources and applications.

CA Education Services

Computer Associates Global Education Services (CA Education) offerings include instructor-led and computer-based training, product certification programs, third-party education programs, distance learning, and software simulation. These services help to expand the knowledge base so you are better able to use our products more efficiently, contributing to your greater success. CA Education has been developed to assist today's technologists in everything from understanding product capabilities to implementation and quality performance.

Because the vast community of education seekers is varied, so too are our methods of instruction. CA Education is committed to provide a variety of alternatives to traditional instructor-led training, including synchronous and asynchronous distance learning, as well as Unicenter simulation.

For training that must be extended to a wider audience—for a fraction of the cost and logistical hassle of sending everybody away to a class—CA Education offers excellent distance learning options.

Computer Associates: The Software That Manages eBusiness

The next generation of eBusiness promises unlimited opportunities by leveraging existing business infrastructures and adopting new technologies. At the same time, extremely complicated management presents challenges—from managing the computing devices to integrating and managing the applications, data, and business processes within and across organizational boundaries. Look to CA for the answers.

CA has the solutions available to help eBusinesses address these important issues. Through industry-leading eBusiness Process Management, eBusiness Information Management, and eBusiness Infrastructure Management offerings, CA delivers the only comprehensive, state-of-the-art solutions, serving all stakeholders in this extended global economy.

For More Information

After walking through this *Getting Started*, you can refer to the numerous resources available to you for additional information. Your product CD contains useful instructional documents that showcase your software and provide detailed explanations about the product's comprehensive, feature-rich components. In addition, the online help system at esupport.ca.com offers procedural information and answers to any questions you may encounter.

Chapter 2

Harvest: Your Key To Project Control

The Basic Methodology

The use of Harvest is divided into administrative tasks and day-to-day tasks. The administrative tasks are performed primarily when first implementing Harvest, and are typically performed by someone designated as a Harvest Administrator. Such tasks include setting up the repository, defining the life cycles to be used, setting up user groups and setting access permissions.

After a Harvest project is set up, day-to-day operation is straightforward. Users are given access to information, provided a set of processes that can be performed and notified of actions, which have been or need to be, taken.

When you have to maintain multiple releases of the same application, Harvest provides you with the tools that you need. For example, you can easily create a "project" for emergency maintenance and keep those changes separate from any ongoing changes in another release. The integrated merge facility enables you to automate the merging of some or all of those changes into a subsequent development process, eliminating labor-intensive manual merges. Both short-term and long-term projects can be developed in parallel, without inadvertently affecting one another.

The concurrent development feature of Harvest allows more than one developer to work on the same area of code simultaneously without fear of overwriting another's changes. Merge utilities are used to resolve any conflicts between the versions.

Harvest is a client/server application. The client and server portions can both be executed on the same machine or be distributed across multiple platforms. The client portion of Harvest consists of the graphical user interface (GUI) and the command line interface. For the UNIX platform, the client portion consists of the command line interface. In addition to the command-line support, users can also use the Harvest Web Client to perform CM functions on the UNIX platform. The server portion contains most of the program logic.

Harvest's Main Advantages

Adaptability: Harvest helps you create and modify models of your own development life cycle and processes through simple point-and-click operations. It then uses this life cycle model to keep software changes under control, schedules on track, and everyone up to date. As changes are made to the development process, the model can easily be updated to support the changes.

Usability: The Harvest user interface is implemented using state-of-the-art GUI technology. It has been designed with ease of use and minimal training time as primary criteria. Icons and toolbars are used throughout the interface, and keyboard accelerators are available for most common functions. Complete online help is also available.

Concurrent Development Support: The concurrent development feature of Harvest allows more than one developer to work on the same area of code simultaneously without fear of overwriting another's changes. Merge utilities are used to resolve any conflicts between the versions.

Parallel Development Support: Harvest makes it possible to maintain multiple releases of the same application. For example, you could work on emergency fixes for a release while at the same time working on the next major version of that product. If necessary, changes made in the emergency area can be merged into the main product development cycle.

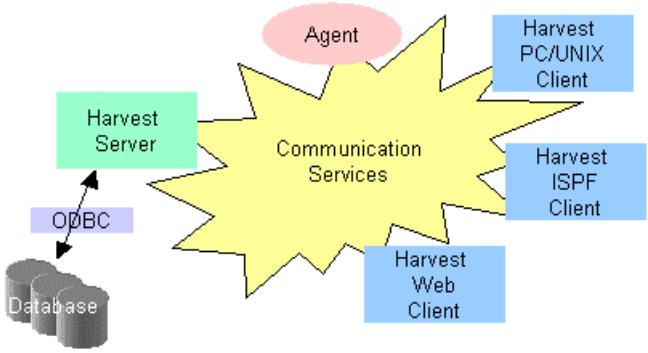
Problem Tracking: Harvest can automate and track problems with associated change packages and forms so that your application production environment always has a history of specific changes that occurred, and the precipitating events that took place within the development process.

Business Workflow Support: Harvest helps you to create and modify models of your own business workflow processes through simple point-and-click and drag-and-drop operations. It then uses your model to keep software changes under control, schedules on track, and everyone up-to-date. By automating the workflow, many routine tasks are also automated, including notifications, approvals, and change migrations from one phase to another.

Tool Integration: Interfaces from popular integrated development environments (IDEs), such as Visual Basic, Visual C++, IBM Visual Age for Java and IBM WebSphere, allow developers to perform routine CCM functions without leaving the IDE.

Complete Solution: All functional groups involved in the development and maintenance process can benefit from Harvest, not just programmers. The entire development process can be controlled, including problem tracking, change management, builds, testing, quality assurance, documentation, and release. Extensive management reporting capabilities support auditing functions throughout the entire process.

Open Architecture: Harvest's open architecture allows easy access to CCM information. Rather than developing yet another database standard, control information is stored within a commercially available relational database. Harvest table formats are fully documented. Database information is normally accessed from the GUI, but a site can access the database directly to generate reports or integrate with other development tools.



The Open Architecture

Harvest's n-tier client/server architecture provides uniform availability of data across the enterprise; it minimizes network traffic, and provides platform-independence to allow an organization to focus on business changes.

This client communicates over TCP/IP from any platform to the central Harvest server, which can be on any UNIX or Windows machine in the network. The server handles all processing and manages access to the CCM repository that can also be distributed to another machine in the network.

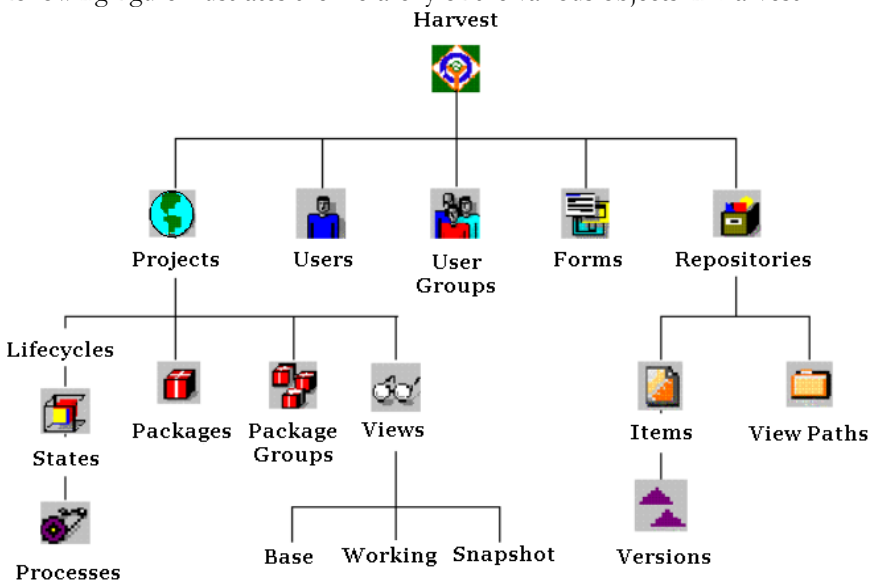
With this architecture, Harvest is designed to provide ultimate levels of scalability to support growing development teams, across widely dispersed geographic locations.

The Version Control Interface (VCI) allows Harvest to integrate with other development tools. For example, you can get access to Harvest version control functions through the Microsoft Visual C++ IDE.

Tip: For more information regarding Harvest's architecture, refer to the product's *User Guide*.

Implementing Harvest

The following figure illustrates the hierarchy of the various objects in Harvest.



Tip: More in-depth discussion can be found in the *Administrator Guide* and the *User Guide*.

Each Harvest object is briefly introduced below. Harvest level objects are global. This means they are available in all projects defined within a Harvest installation:

Broker—A program in the communication layer of Harvest that provides access to your local machine. The broker assigns an appropriate server to each local machine.

Project—The project represents the entire control framework within Harvest that supports a particular development or maintenance activity.

Users—These are the individuals defined in Harvest.

User Groups—Individual users can be assigned to one or more user groups for the purpose of access control and process setup.

Forms—Harvest forms can be used in a similar way as paper forms are used. For example, they can be used to track issues and problems, or as a structured method of communication.

Repositories—Harvest maintains the data under its control in one or more repositories. A repository can be associated with more than one project.

Project level objects are specific to a single Harvest project. These include States, Packages, Package Groups, Views, items, items paths and the objects that derive from them.

States—States are distinct work areas in which certain activities can take place as packages move from identification to completion. A life cycle can include any number of states.

NOTE: A *process* is an action that can be executed on an object in Harvest. The processes defined for a state determine what activities can be performed in that state.

Packages—A package is the basic unit of work that moves through the life cycle. It typically represents a problem that needs to be tracked and any associated changes made in response to the problem. Packages can be assigned to a user and are moved from one state in the life cycle to another, based on user approval.

Package Groups—A project can have many packages. A package group allows for logical grouping of related packages. A package can belong to one group or several, or it might not belong to any group.

Items—An item refers to a component of a repository. This is to distinguish between data outside the repository and data within it.

Versions—Versions are created when changes are made to an item. A version represents the initial item and all changes associated with previous versions.

Views—Views specify the item versions visible to a state. Base views represent the initial starting point of a project life cycle. Working views isolate functional work areas. The working view determines which item versions are accessible to users. Snapshot views are read-only images of a working view at a specific point in the project life cycle. A snapshot view can be used to support the baseline (base view) of a new project life cycle.

View Paths—Directories within the repository are referred to as view paths to distinguish them from directories in the external file system.

The term project refers to the entire control framework within Harvest that supports a particular development or maintenance activity. A Harvest project has a "life cycle" which describes and controls the flow of activities that take place within the project. The project life cycle consists of states, processes, views and access settings.

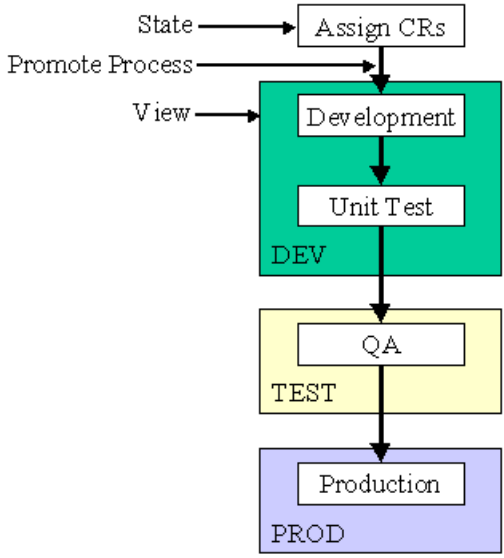
Analyzing your current software development and maintenance activities and mapping them into a life cycle is a crucial first step to implementing Harvest successfully. Consider the following questions and plan your life cycle before attempting to actually set it up in Harvest.

- How many functional groups are involved in the life cycle? Of these groups, which need to have their activities isolated from one another and which can share a common working area?
- How many separate phases of development can be identified? At what point are approvals required and from whom? Who should be notified when a change is ready to move from one phase to another?
- What activities take place in each separate phase of development, and who needs to do them?
- How will problem tracking be used in conjunction with development? Will problems have their own life cycle in a support project, or will they be created in a development project?

The following is the typical sequence for defining a new project life cycle:

1. Create the Harvest project.
2. Define the project baseline.
3. Create Users/User groups.
4. Define access control for the various project objects.
5. Refine the life cycle processes needed for change management functions.
6. Perform package management functions: package assignee and form association.

The following figure illustrates some of the basic building blocks used to create a Harvest life cycle:



In this example, there are five states. Assign CRs is the state in which packages are created to fix change requests. Development is the state in which developers modify code, and Unit Test is a state in which they test their changes. QA is the state in which QA personnel test the application and Production is the state that holds code ready to be released.

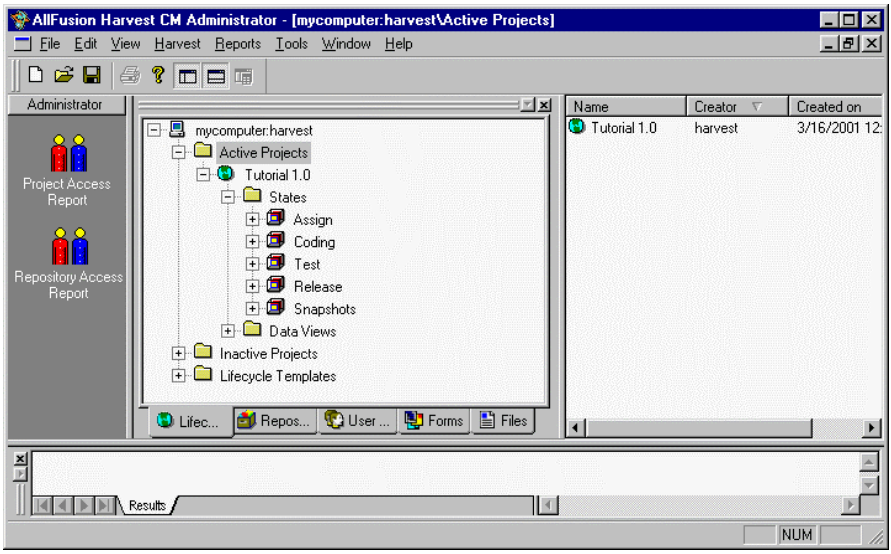
The initial state, Assign CRs, is where the initial change request is generated as a package/form association. The Assign CRs state is not associated with a data view because no changes will be made in this state. For example, the CR package can be promoted to the Development state, which shares the DEV data view with the Unit Test state.

The changes made to a package in the DEV view are not visible in any other state that does not share the same view. In this example, after a package has completed testing in Unit Test, it is promoted to QA, which is in the TEST view. The TEST view is then updated with the package changes. Changes made in the DEV view by other packages do not appear in TEST until those packages are promoted to QA.

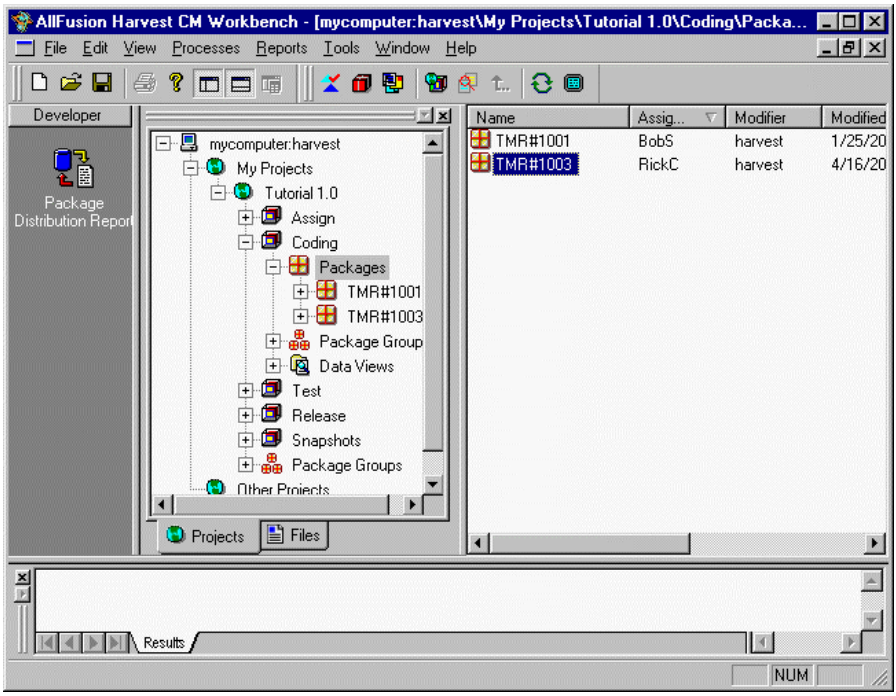
Main Window Features

After logging in to Harvest, the main window displays. The main window depicts Harvest objects and their relationships and provides access to functions. The window is divided into four panes: the taskbar on the left, the workspace in the middle, the document view on the right and the output log across the bottom. The status bar is the horizontal area at the bottom of the window and provides information about the current menu you have selected in the main window.

Harvest Administrator View:



Harvest Workbench View:



The four panes are:

Taskbar—The taskbar is the left pane on the main window. If you have favorite or frequently used tasks (scripts or reports) to run, you can create a shortcut icon for each one on the taskbar. The taskbar can be divided into groups of shortcuts to help organize your information. You can add and remove groups, shortcuts, change the icons and hide the taskbar.

Workspace—The workspace shows all Harvest objects to which you have access. Your Harvest broker is depicted as a workstation and is identified by broker name. The broker icon is located at the top of the hierarchical tree in the workspace.

If you log into the Harvest Administrator window, the workspace will consist of five tab pages: Lifecycles, Repositories, User Groups, Forms and Files. If you log into the Harvest Workbench, you will see only the Files and Projects tabs. The objects within each tab, with the exception of Forms, are displayed in a hierarchical format.

List View—The list view displays information related to an object that is selected in the workspace. For example, if you select a user group in the workspace, the users that belong to that group will be displayed in the list view.

Output Log—Most activities in Harvest generate output. This output can be the focus of an activity, such as in generating reports, or it is informational, documenting the results of an action. Output is displayed in the log at the bottom of the main window. Information in the output log can be printed, copied or saved to an external file.

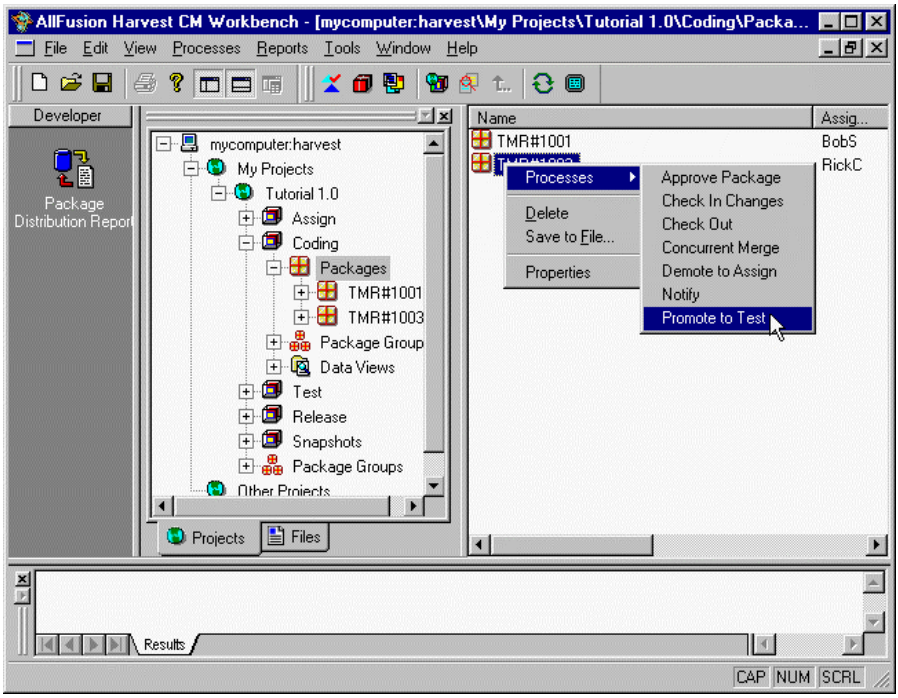
Tip: When logging in as an administrator, the main window is called the Administrator window and provides functionality suitable for an administrator. When logging in as a user, the main window is called the Workbench and provides functionality suitable for a user.

Installation

Tip: Please refer to the *Install Guide* for information on installing the client and server.

Using The Process Menus

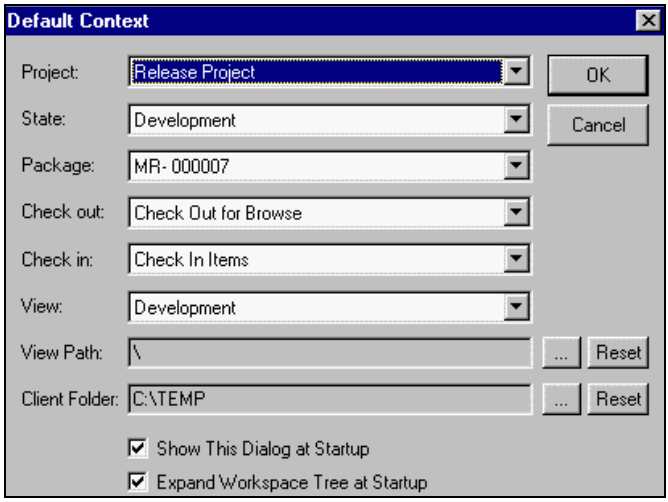
Menus on the main window provide you with access to functions and processes. For example, you can also right-click an object in the workspace or list view to invoke a shortcut menu, which includes available processes and the properties dialog for the object. Keyboard shortcuts appear on the menus to the right of many commands.



Toolbars

The toolbar at the top of the screen contains buttons that offer convenient and fast access to day-to-day Harvest functions.

From the toolbar, easily set your Harvest working context: Project, State, Package:



Project—The Project field can be set to automatically populate with your current project location. To change the project, use the drop-down list to choose a destination project.

State—The State field can be set to automatically populate with your current state location. To change the state, use the drop-down list to choose a destination state.

Package—The Package field can be set to automatically populate with your current package selection. To change the package, use the drop-down list to choose a package.

Check Out—Choose a default check out process from the drop-down list. The list shows all check out processes defined for your current state.

Check In—Choose a default check in process from the drop-down list. The list shows all check in processes defined for your current state.

View—The View field is automatically populated based on the selection made in the State field. Once the State has been selected, the View associated with that State is automatically selected. If the State has no view, then this field will remain blank.

View Path and Client Folder—The View Path and Client Folder fields work together to help you synchronize the internal and external file structures. They are typically used to establish a point at which paths in the repository mirror working directories on the client. The default View Path and Client Folder context settings can be overridden at process execution time.

View Path—The view path specifies a location in the repository from which items will be checked out or files will be checked in. By specifying a view path, you are setting a default path that will be placed in this field in subsequent check outs and check ins. To change your repository path, click the button next to the View Path field. This invokes the Select a Repository Item Path dialog that allows you to browse through available paths and select a location.

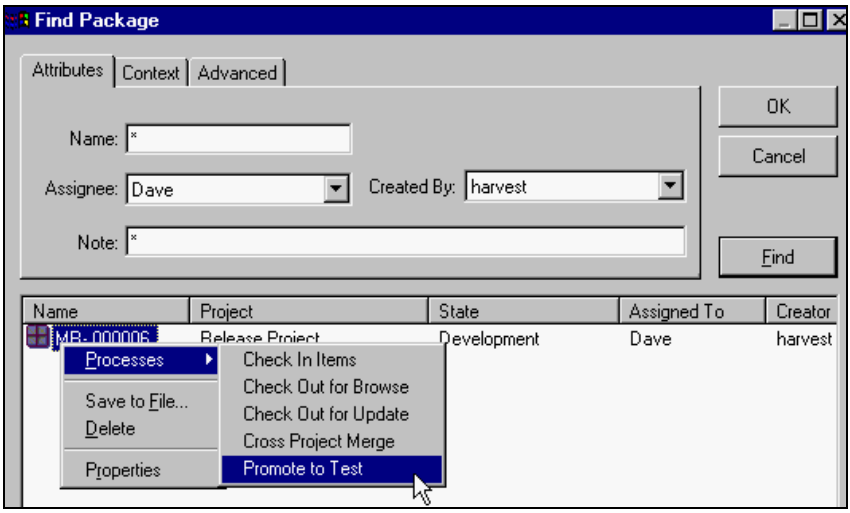
Client Folder—The directory selected for the Client Folder field specifies the destination on the file system from which files will be checked in or to which items will be checked-out. If either of the preserve structure check out options is being used, the check out specifies a path and this specification is appended to the file system directory. To change the destination, click the button next to the Client Folder field. This invokes the Select a Directory Path dialog that allows you to navigate through directories available on your system to select a file.

Show This Dialog at Startup—By default, this option is enabled. After logging in to the workbench, the Default Context dialog will display. If you prefer logging directly into the workbench, you can thereafter bypass the dialog by disabling this option.

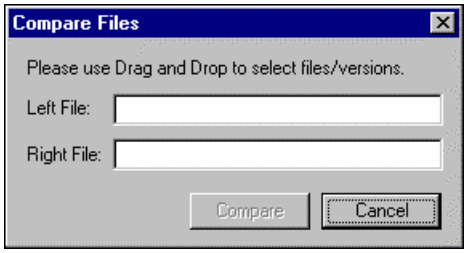
Expand Workspace Tree at Startup—By default, this option is enabled. This option causes the workspace tree on the workbench to be opened to the context you specify in this dialog, after you log in.

The Find utilities enable you to locate Harvest objects using specific filtering criteria. Using Find Form, Find Package or Find Version generates a list of objects. You are then able to execute processes by right-clicking these objects and selecting one of the available processes.

Here is an example of finding a package:



Compare two files or Harvest versions by clicking the Compare Files icon (two squares with the magnifying glass, in the top menu bar) and using the drag and drop utility to populate the Left File and Right File fields. Click Compare to generate a visual difference report.

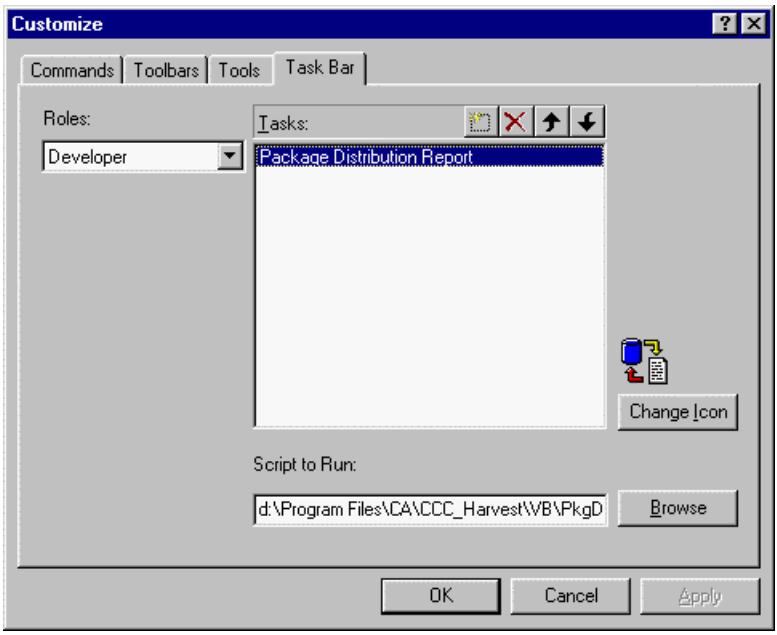


Taskbar

The taskbar is the left pane on the main window. From the taskbar you can create shortcuts to launch and interpret Visual Basic and Java scripts. The taskbar can be divided into groups of shortcuts to help organize your information. You can add and remove groups, shortcuts, change the icons, and hide the taskbar. You may also:

- Click once on a shortcut on the taskbar to run a task without exiting the main window.
- Change groups to move to a different set of shortcuts.

You can set up the taskbar by using the Taskbar dialog. The Taskbar dialog is available from the shortcut menu on the taskbar or by choosing Tools, Customize and clicking the Taskbar tab.



Roles—The name of your current group is shown in this drop-down list. You can choose a different group by choosing its name from the list.

Tasks—Using the Tasks field and toolbar, you can add tasks, delete tasks, and arrange the order of tasks in your taskbar.

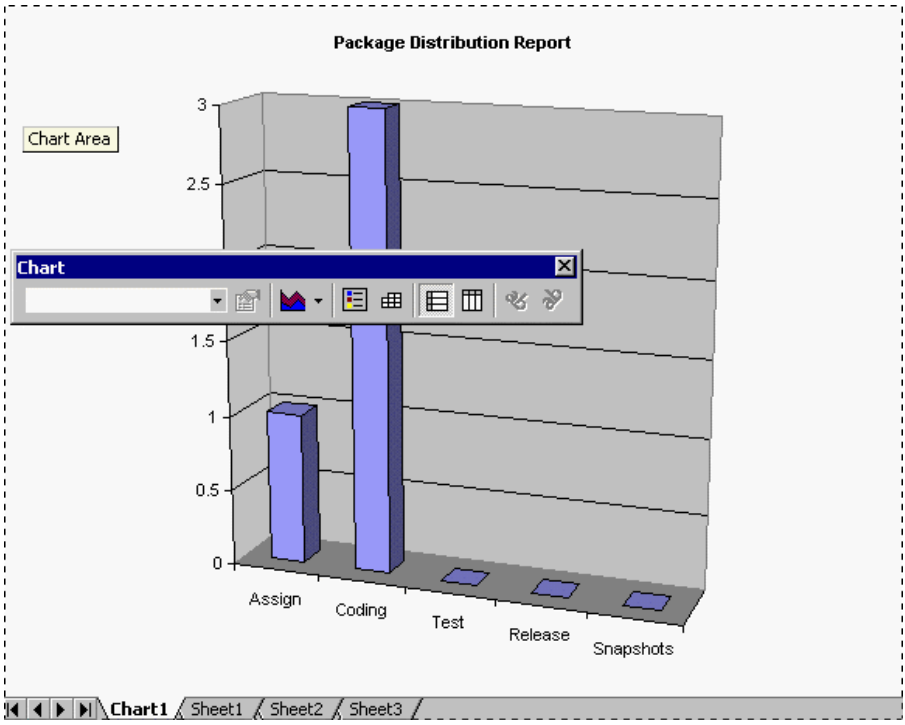
- To add a task, click the toolbar New button. A new task field appears in the task list. Type a name for the task in this field.
- To delete a task, select the task name in the list, and then click the Delete button on the toolbar.
- To arrange the order of the icons in your taskbar, select the task name in the list, and then click the toolbar Move Up or Move Down buttons.

Script to Run—You can locate and specify a program to run for your task, by clicking the button next to the Script to Run field to open the Windows Explorer file chooser.

Package Distribution Report

From the taskbar on the workbench, you can generate a report showing the location of packages in a project by clicking once the Package Distribution Report icon in the taskbar. A Microsoft Excel report displays in the list view. A functional Excel toolbar is added to the workbench and removed when you close the report. The menu bar provides options to save, print, and so on.

NOTE: Set your context before generating the report.



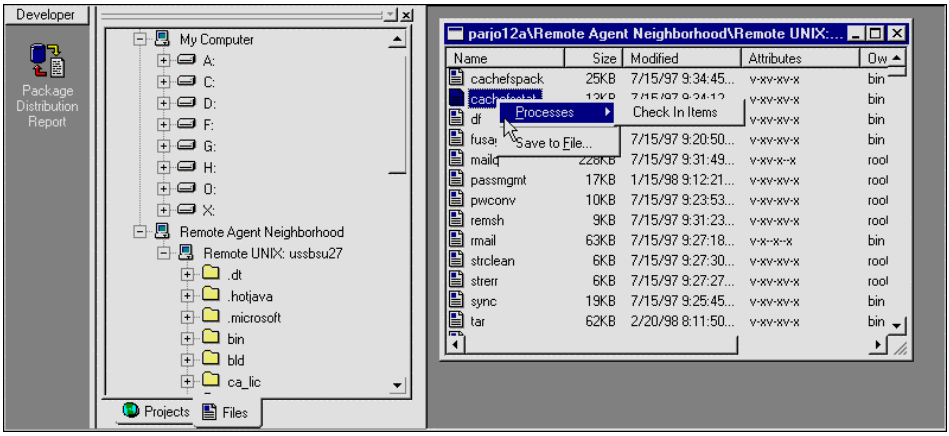
Access Local Or Remote Filesystems

The Files tab displays all of the files that you can access on your client workstation.

- The *My Computer* folder shows the drives that are mapped to your local machine.

- The *Remote Agent Neighborhood* folder shows all available remote file agents. Using a remote file agent, you can check in files from remote locations.

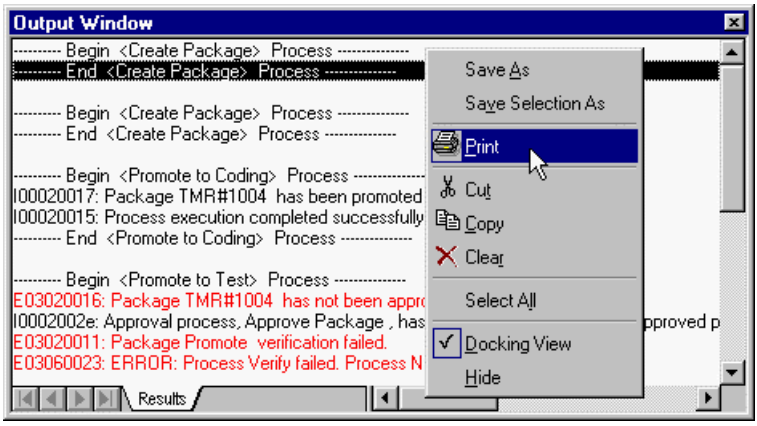
To easily locate an agent in the workspace, you can place it in your neighborhood. The agent(s) will remain in your neighborhood until you remove it.



Output Log

Most activities in Harvest generate output that is sent to the output log. This output can be the focus of the activity, such as in generating reports. At other times, it is informational and documents the results or consequences of a previous action. Error messages are shown in red text for high visibility.

Whenever new information is sent to the output log, it is appended to the end. Its contents are maintained throughout the session, unless it is cleared or closed. Information in the output log can be cleared, copied, printed, or saved.



What's Next?

Harvest can easily adapt to your company's standards, requirements, and procedures. Its solutions-oriented structure addresses your most common development organization requirements. Next, you will learn how this product will help you manage your organization. Your employees will know where they fit into the general scheme of your business, and how their work and responsibilities affect other members of your team. Let Harvest lend your business a hand.

Chapter 3

Create Teams Within Your Organization!

Configure Harvest To Your Needs

Now that you have covered the Harvest methodology, you are prepared to indulge in the great features of this product, whether you are experiencing Harvest for the first time, or you are upgrading from a previous version!

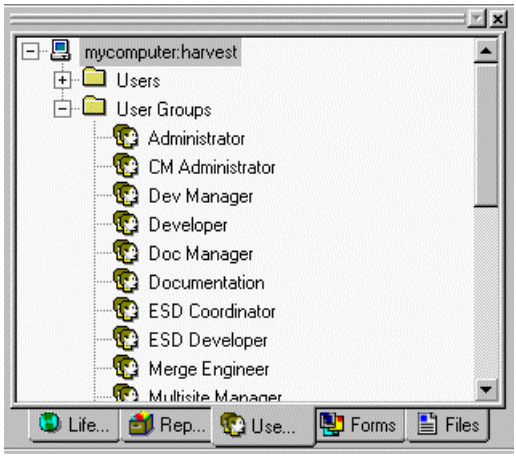
The use of Harvest is divided into administrative tasks and day-to-day tasks. The administrative tasks are performed primarily when first implementing Harvest, and are typically performed by someone designated as a Harvest Administrator. After a Harvest project is setup, day-to-day operation is straightforward.

Creating User Groups

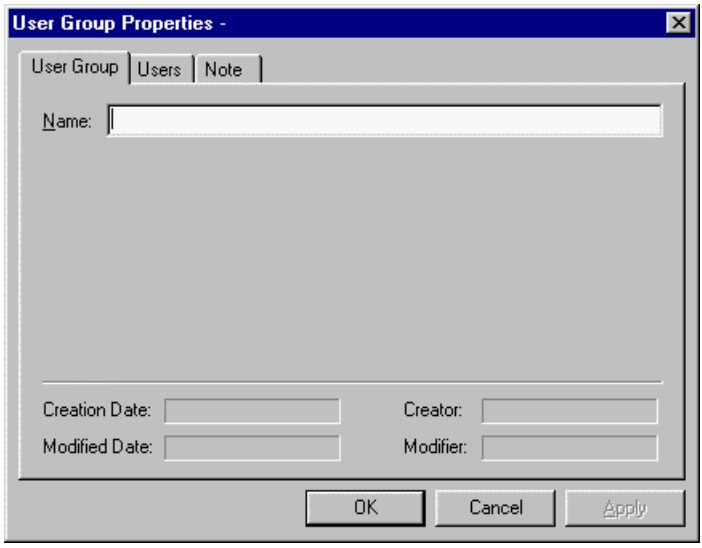
It is now time to organize your business by formulating Harvest to your needs. With Harvest, you will create User Groups and assign Users to each of these groups. Users will have access to the information they need in order to complete their daily tasks, and you can save yourself many headaches, knowing that the day-to-day operation of your business will run smoothly.

To create User Groups, follow these simple steps:

1. Select the User Groups tab  at the bottom of the workspace. You will see two folders: Users and User Groups.



2. Right-click the User Groups folder and choose *New User Group* from the shortcut menu. This will display the User Group Properties dialog.



3. Enter a name for the User Group.
4. Repeat steps 2 and 3 for each User Group you wish to import, and then click OK.

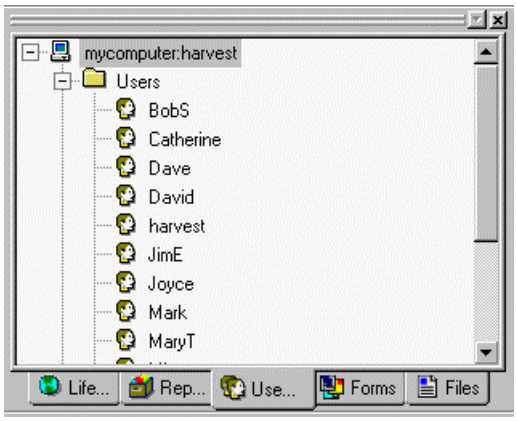
Creating Users

Now that you have a list of User Groups, you will assign Users to these groups. With these User Groups, you will be able to organize your employees by grouping them into specific categories. This will enable you to manage your business more efficiently, and will, at the same time, give your organization and its employees, a structured environment to work within.

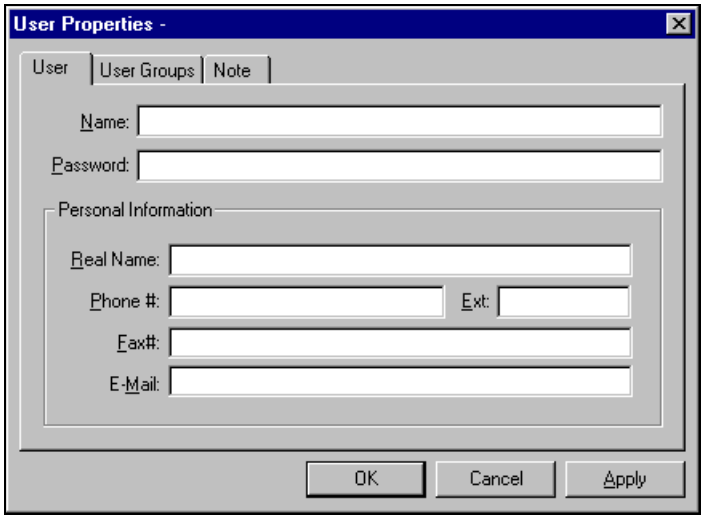
NOTE: Harvest provides Administrators with a ready-made list of User Groups. Users can be added to these existing User Groups or new User Groups may be created to further customize Harvest to your requirements.

To create Users, follow these easy steps:

1. From the User Groups tab  select the Users folder.



2. Right-click and choose *New User* from the shortcut menu. The User Properties dialog will appear.



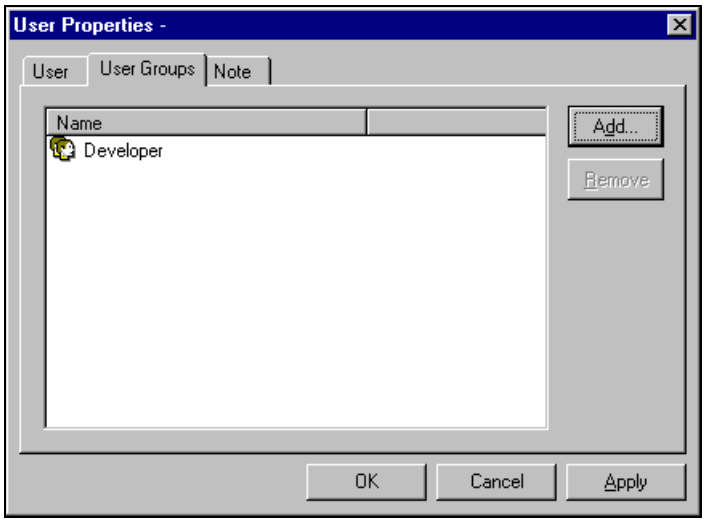
3. Enter a user name in the Name field. This will become the login name for this User and identifies him or her throughout Harvest.

4. Enter a password.

NOTE: The Personal Information section is not required to define a user.

If you want to add the User to an additional User Group:

1. Click the User Groups tab on the User Properties dialog.
2. Click *Add*. A list of the available User Groups (those which you specified previously) is displayed.



3. Select the User Group you want to add the User to, and click *OK*.
4. Click *OK* to close the User Properties dialog.

Tip: To create additional users and add them to a User Group, right-click the Users folder and choose *New User* from the shortcut menu. After filling out the user name and password, simply repeat the steps above for adding them to the User Groups you prefer.

Creating A Project

Now that you have selected Users and assigned them to particular User Groups, you will now create a project. A project refers to the entire control framework within Harvest that supports a particular development or maintenance activity. A project includes information about what data to access, how changes progress through the cycle, the activities that can occur, and user responsibilities.

In Harvest, projects consist of the following objects: states, processes, packages and package groups, views, repositories and items, and versions.

With Harvest, you can have many different projects, depending on the applications being controlled and the kind of development activity undertaken. For example, there might be one project for maintaining an already released version of an application, another for developing the next release and a third for maintaining code shared among various applications. There might be an entirely different project used by the support group for tracking incidents and problems.

Harvest provides ready-made life cycle models that you can use when setting up your configuration management system. These models provide solutions for software development, as well as for problem tracking and general workflow scenarios.

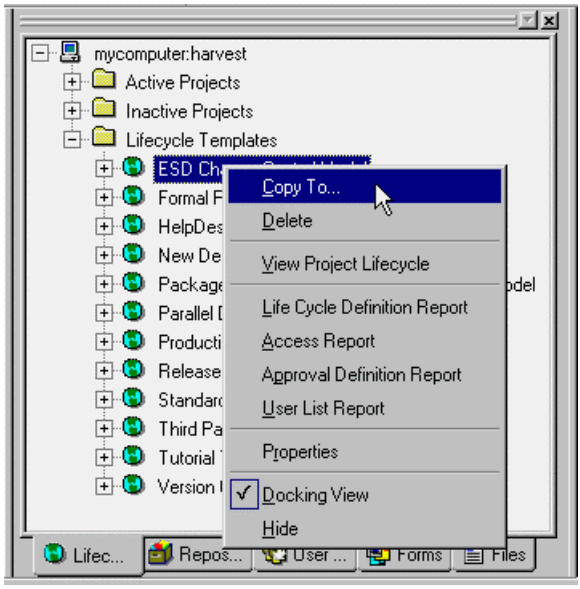
Using the Harvest workbench, you can perform most of your day-to-day Harvest activities, such as creating a package and form, checking items in and out of version control, and moving packages through the life cycle.

NOTE: Information about creating and using packages and forms can be found in Chapter 4 in this guide.

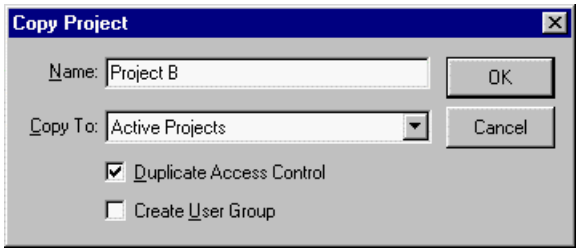
In order to create your Harvest project, follow these simple steps:

Tip: When creating a project, it is recommended that you base your project on an existing one, or a life cycle template, by copying it and then editing it.

1. Click the Lifecycles tab  at the bottom of the workspace, and then open the Lifecycle Templates folder.
2. Right-click a life cycle template and select *Copy To*.

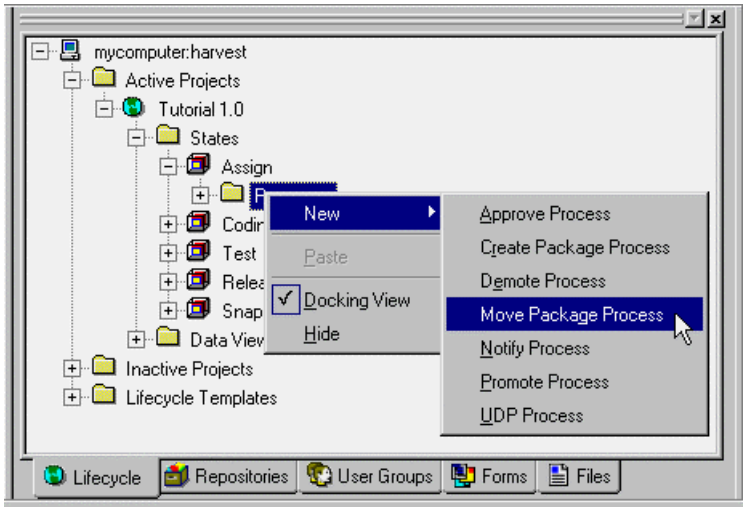


3. Give the project life cycle a new name and copy it to the Active Projects folder.

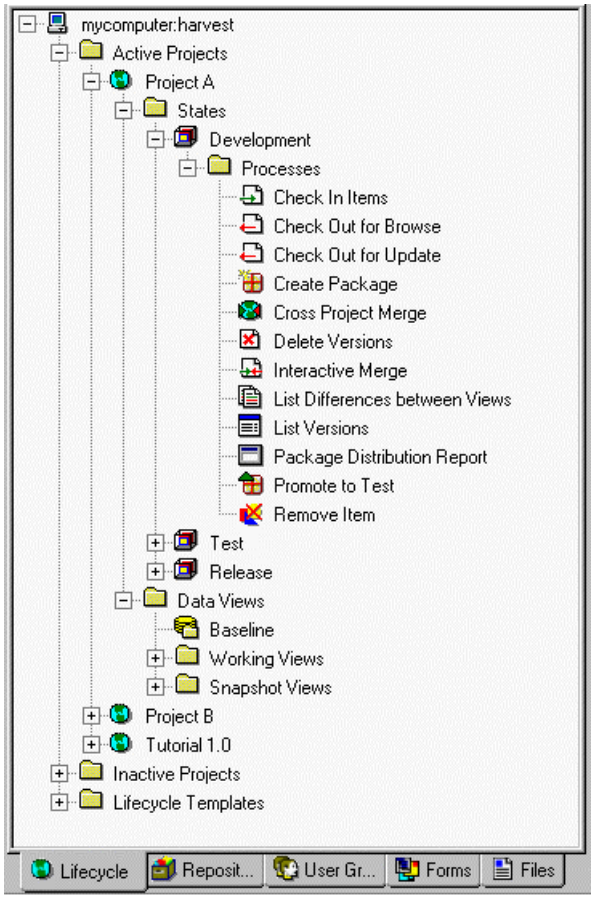


4. To duplicate an initial access control setup, place a check mark in the "Duplicate Access Control" checkbox. The "Create User Group" option creates a project-based User Group to assign an existing User Group to.

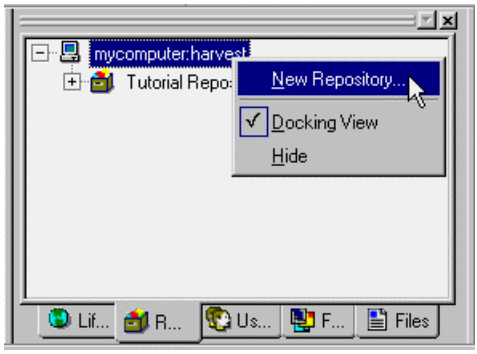
   The life cycle can be refined easily. For example, additional processes can be added to a state:



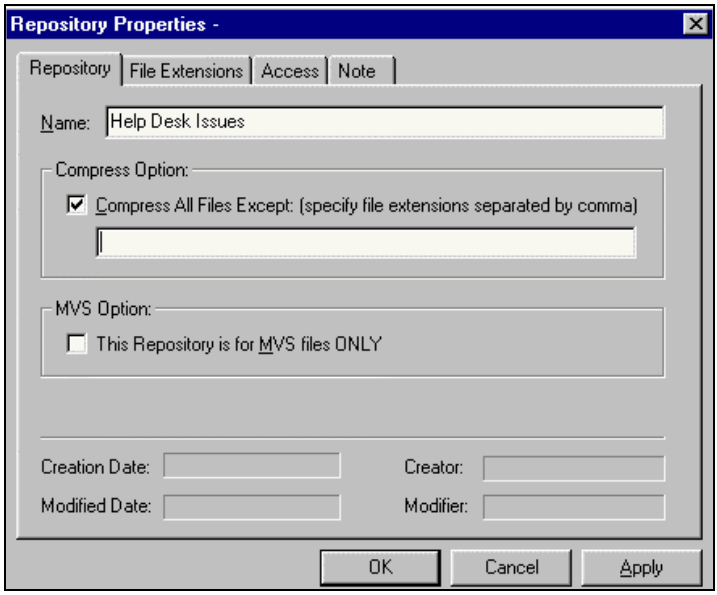
5. The duplicated project is a complete copy of the life cycle template. If you expand the project folder and continue to navigate down the project structure, you will see folders for States and their processes as well as Data View, Working Views and Snapshot Views folders.



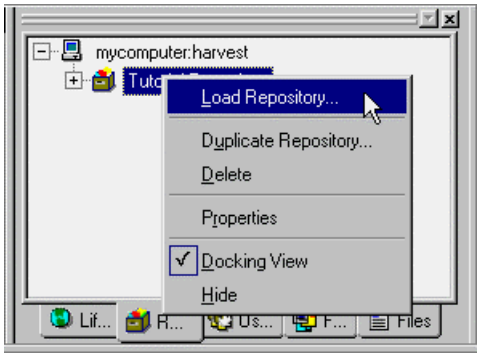
6. Select the Repositories tab  and right-click the broker name to create a data repository.



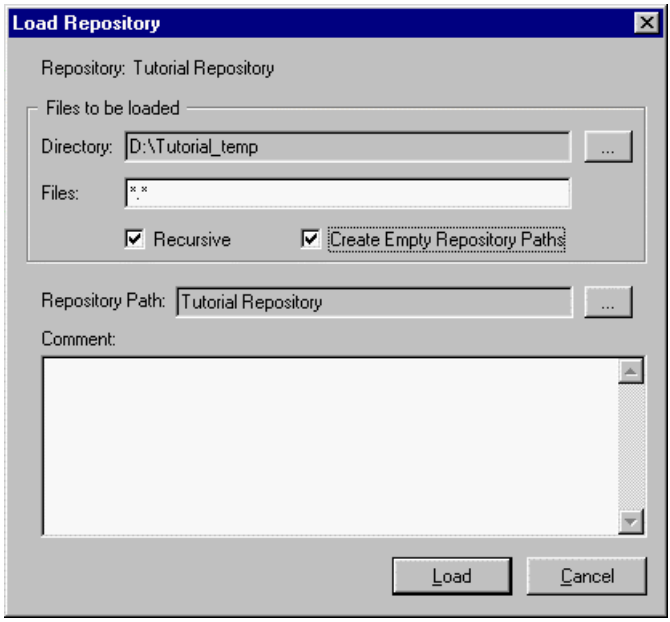
The Repository Properties menu allows you to define the name of the repository. The compression option saves database storage space and improves performance.



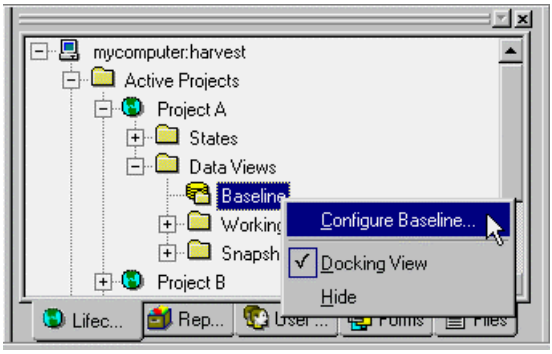
7. Right-click the repository to load the data.



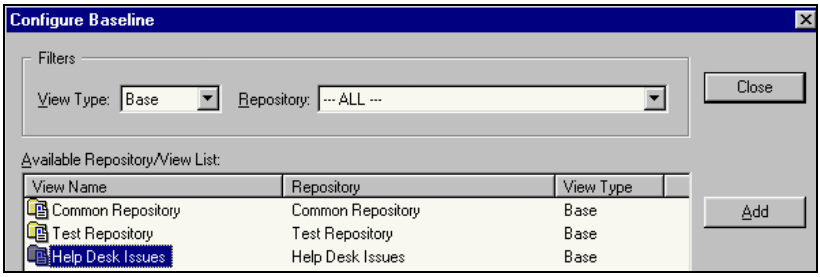
8. Select a top-level directory to load. The recursive option will enable all directories and files below the top level to be loaded. Empty directory paths can also be loaded. Select *Load* to load data in the repository.



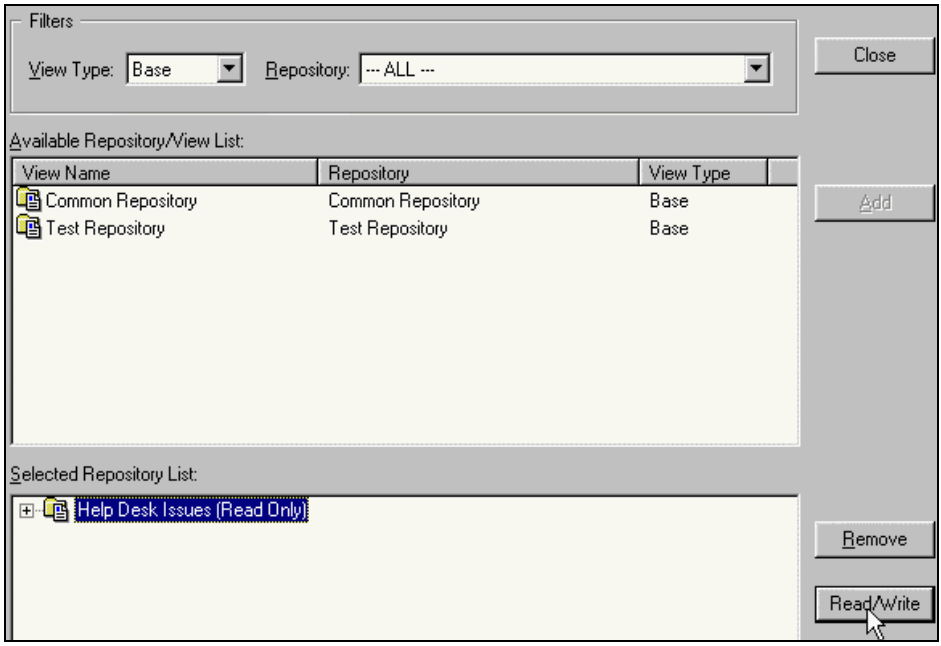
9. Configure the project "baseline." This establishes the starting point in the project life cycle. Select the Lifecycles tab  and expand the Project node, then the Data Views node. Right-click on *Baseline*.



The Configure Baseline dialog appears. The dialog shows the available repositories to add to the current project baseline. You can add the Helpdesk repository by highlighting the repository and selecting *Add*.



The repository is then made into a read-write file so that changes can be made against it in the project life cycle.



10. Click *Close* to close the Configure Baseline menu.

What's Next?

You have just set up Harvest to reflect your organization's specific needs. Next, we'll take a look at combining all the necessary items together to create packages, which will then travel through Harvest's life cycles! Once these packages begin their journeys, they will have to be managed and maintained for optimal use. Harvest makes this easy.

Chapter 4

Management Is The Key To Efficiency

Packages: Your Solution To Managing Change Across The Enterprise

One of the biggest challenges organizations face is how to manage a change process that includes many components, involves countless events and the coordination of numerous people. When events take place at different sites and in different time zones, the opportunities for errors increase dramatically.

Harvest uses the concept of a package to manage the change process across the enterprise. The package is used as the centerpiece for identifying change items, coordinating approval processes, and synchronizing the overall workflow of your organization!

The Basic Unit: Understanding Packages

A package is the basic unit of work that moves through a life cycle in Harvest. It typically represents a problem or a request that needs to be tracked, the changes made in response to the problem or incident, and any other associated information.

A package contains everything that is required to document, manage or implement change. A user can easily determine what went into an application change, because it is all in the package. The package shows who signed off on the change, who requested it, who worked on it, and so on. And, if a change does not work for some reason, the entire package can be easily demoted!

Tip: A promote process moves changes from one state to the next. A demote process returns changes to a previous state. For more information about states, refer to Chapter 2 in this guide.

The life span of a package usually takes place within one environment, or project; however, packages can be created in one environment and then moved to another to begin a new life cycle.

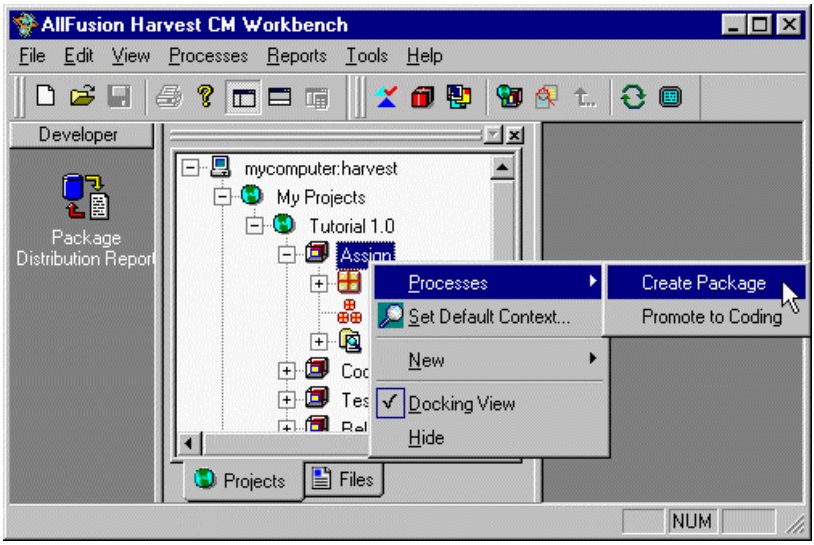
The package is your link to the actual data accessed during the change process. Each package resides within a particular state of the life cycle. Changes are associated with packages during the Check In or Check Out processes.

NOTE: For more information regarding Check In and Check Out processes, please refer to the *User Guide*.

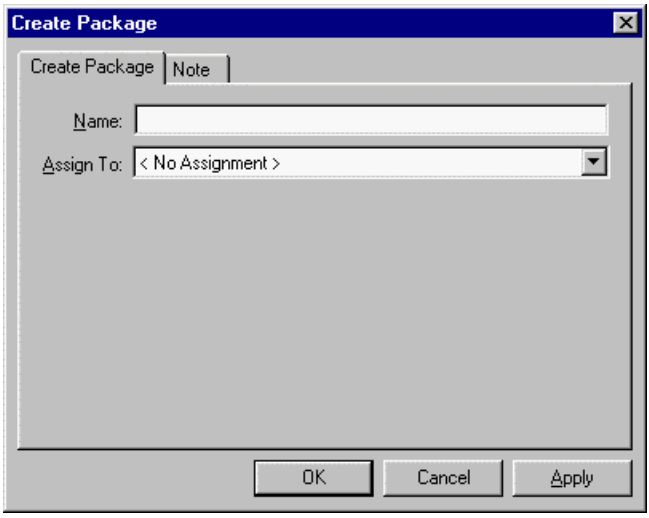
Packages are promoted or demoted between states and become the vehicle used to move associated changes through a life cycle.

To create a package:

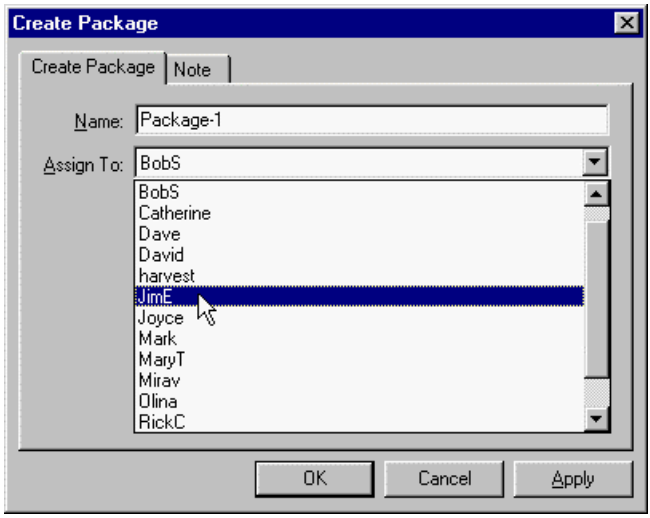
1. In the main window, on the Projects tab  of the workbench, right-click the state in which you want to create a package.
2. Choose *Processes*, *Create Package* from the shortcut menu, or choose *Processes, Create Package,* from the main window menu.



The Create Package Process Execution dialog appears.

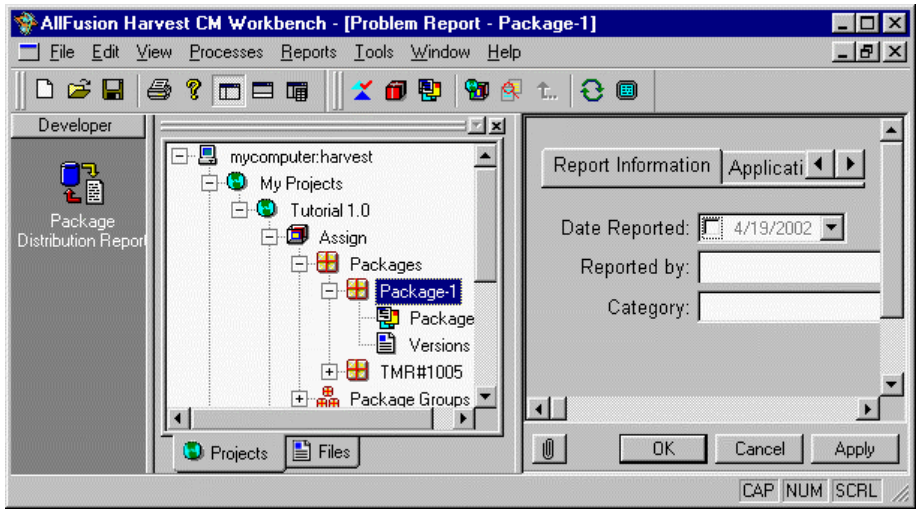


3. Type in the name of the package. Select the *Assign To* drop-down menu to assign a User to this package.



4. Click *OK* to create the package.

   A form will be automatically created and associated with the new package if the *Create Package Process* has been set up to automatically create a form. The form will look similar to the following:



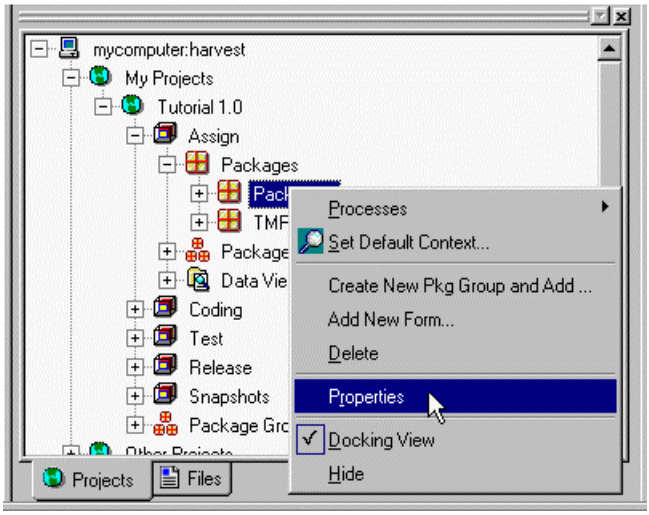
Tip: For more information about Forms, and how they relate to Packages, refer to Chapter 5 in this guide.

Detailed package approval information is available from the package properties.

NOTE: An Approval Process must be defined for the state containing the package to view the approval information. See the *Administrator Guide* for more information.

---

5. On the Workbench, expand the Packages node and right-click the package just created, to access the package properties.



6. Select the Approval tab on the Package Properties dialog.

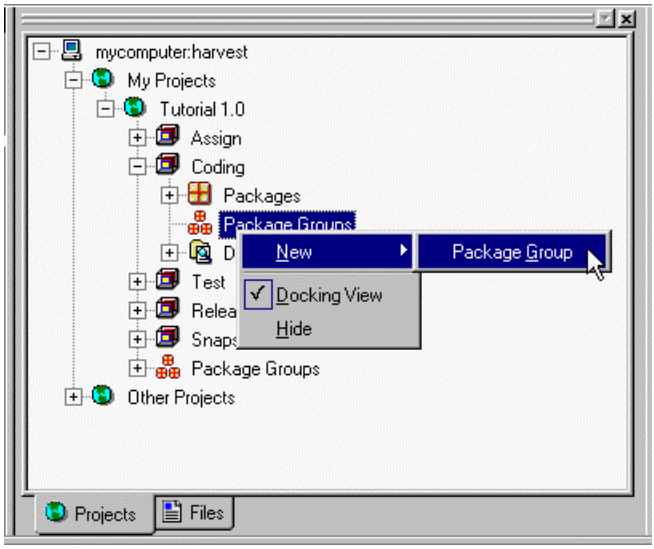
This is the pending approval action status. Waiting indicates that a user approval action is needed. When a package is approved, the Action status changes to Approved.

Creating A Package Group: Part One

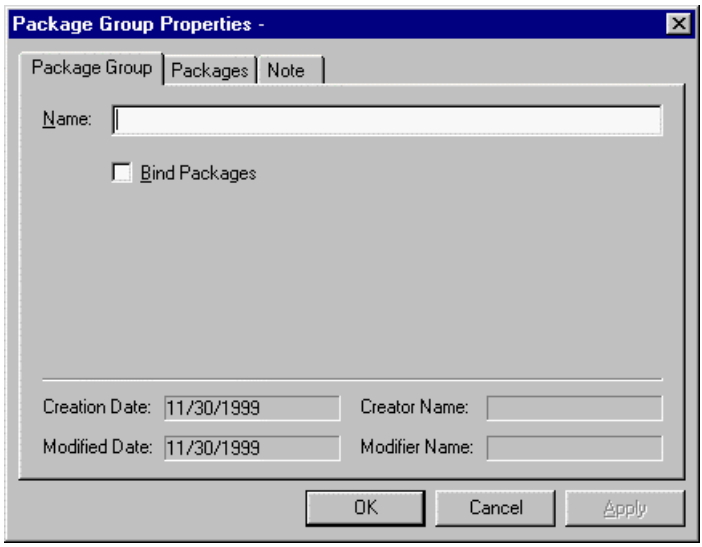
A package group provides a way to operate on related packages as a unit. A particular package might belong to one group or several, or it might not belong to any. Any operation that can be performed on a package can be performed on all packages in a package group. For example, a User can promote all packages in a group from one state to the next. If an approval is required before packages can be promoted, another user can approve the promotion of the group.

To create a package group:

1. On the Projects tab  of the workbench, select the project or state in which you want to create the package group.
2. Right-click and choose *New*, *Package Group* from the shortcut menu.

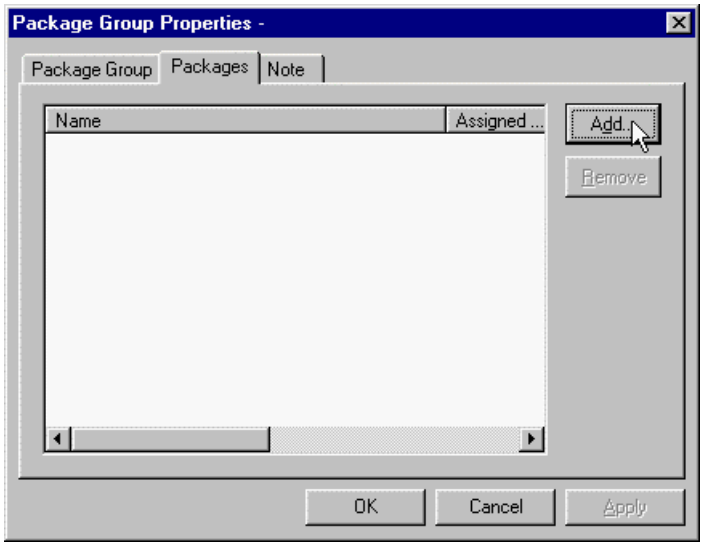


The Package Group Properties dialog is displayed.



3. Give the Package Group a name and click *Apply*. The Package Group Properties dialog will remain active.

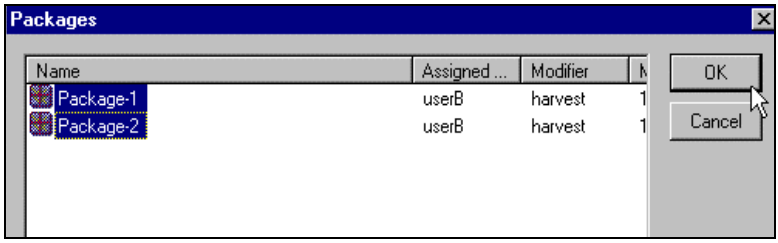
4. Next, add a package to this package group.



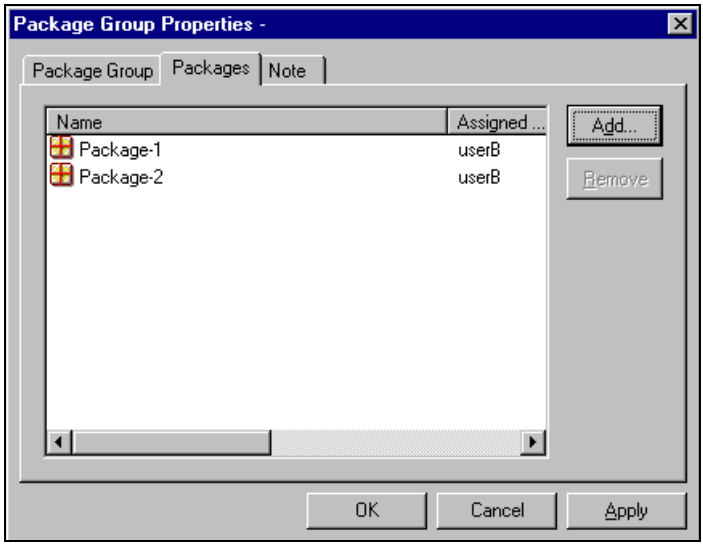
5. Select the Packages tab and click "Add." This brings up a list of Packages that can be added to the Package Group. You may select one or all of the packages listed to add to the Package Group.

NOTE: A Packages list window appears, from which you may select one or all of the packages listed to add to the Package Group.

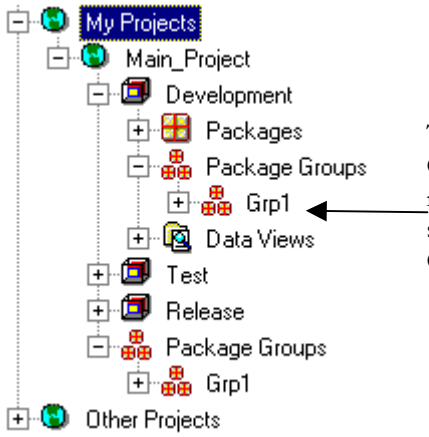
6. Select the chosen packages, and click *OK*.



Clicking OK populates the Package Group with the selected packages.



7. Clicking OK in the Package Group Properties dialog saves the changes made to the Package Group. The newly created Package Group now appears in the Package Groups folder at the state level and at the project level.

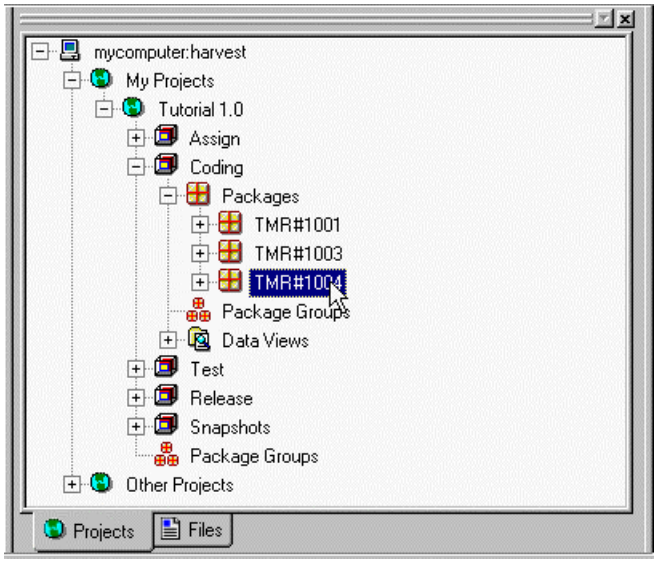


The Package Group, Grp1, which was recently created, is now shown in the Package Groups folder.

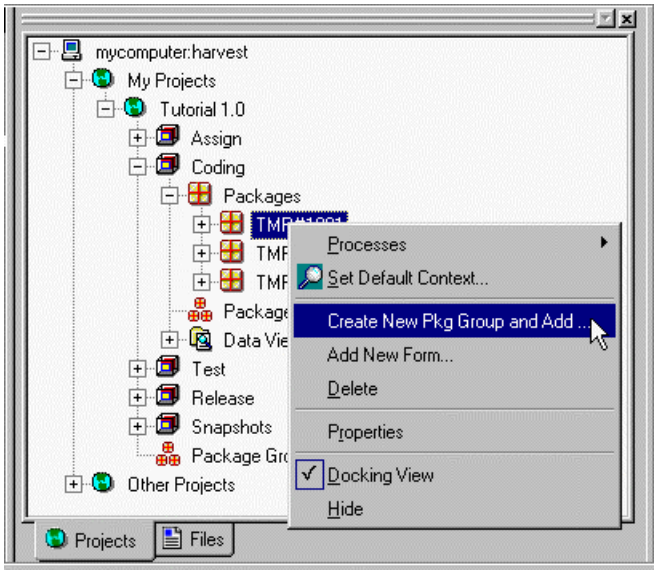
Creating A Package Group: Part Two

Another way to create a Package Group is to create and assign a package to the Package Group at the package level:

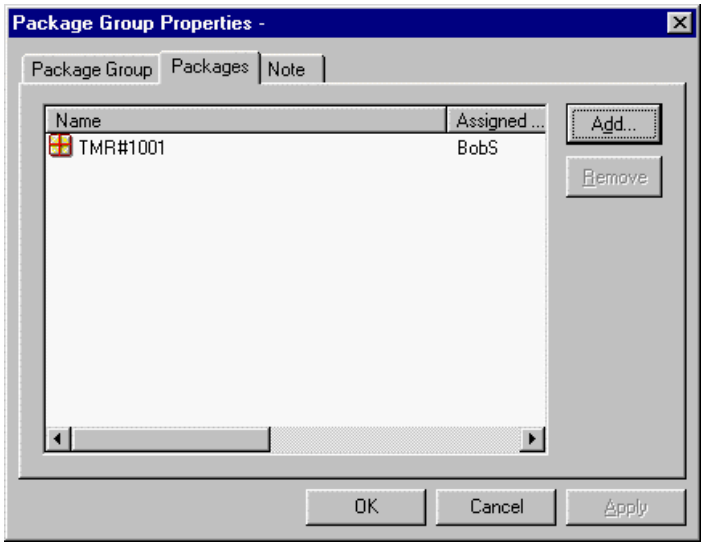
1. On the Projects tab of the workbench, open a Packages folder in a state and select a package that you want to add to a Package Group.



2. Right-click and choose *Create New Pkg Group and Add …* from the shortcut menu.



This brings up the Package Group Properties dialog and allows you to create a Package Group. Note that the package chosen is already added to the Package Group.



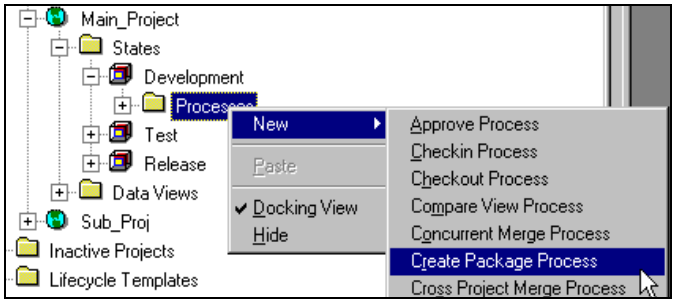
Automatically Creating A Form

As was previously mentioned, when creating a package, a form will automatically be created and associated with a new package if the Create Package Process has been set up.

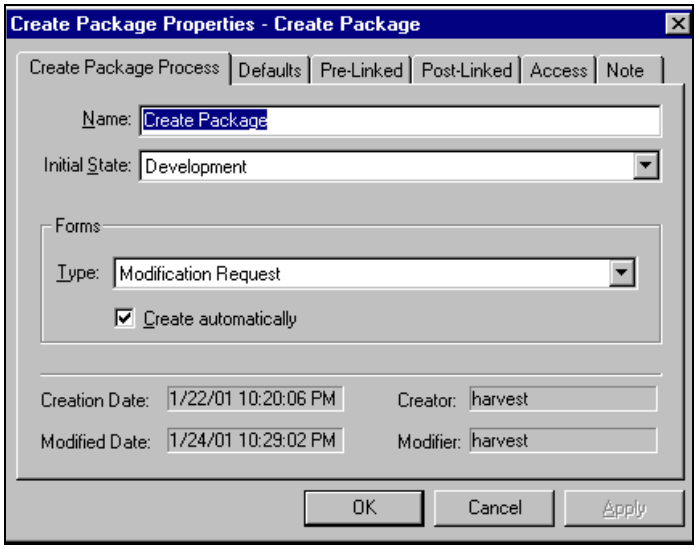
> **Tip:** To learn more about forms and their functions, please refer to Chapter 5 in this guide.

Forms may be generated automatically, so that each time you create a new package, a form will automatically be created as well. Define the Create Package process to automatically create a form:

1. On the Lifecycles tab navigate to the state and then select the Processes folder.
2. Right-click the Processes folder and choose *New, Create Package* from the shortcut menu.

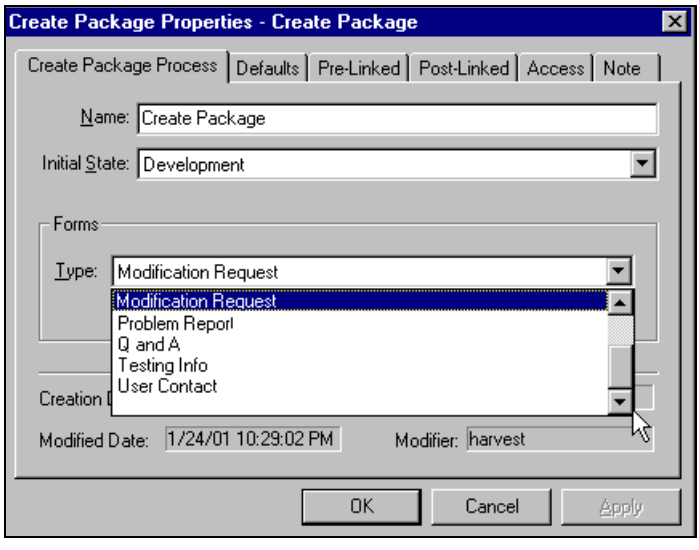


3. This opens the Properties dialog in which you can define the properties of the process.



4. Enter a name for the process, and enter a name for the Initial State. In the Forms section, select the Type drop-down menu. All available forms will be listed here. Make sure that "Create automatically" is enabled.

NOTE: It should be remembered that having a form associated with a package is not required, and, therefore, you do not have to enable "Create automatically."



5. Click OK.

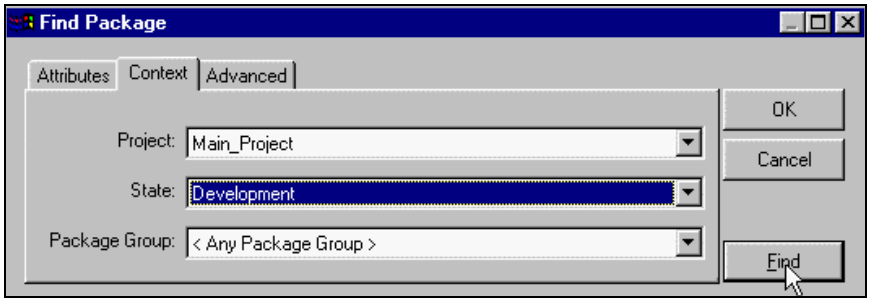
You have now set up Harvest to automatically create a form whenever a new package is created.

Package-Based Check-In/Check-Out

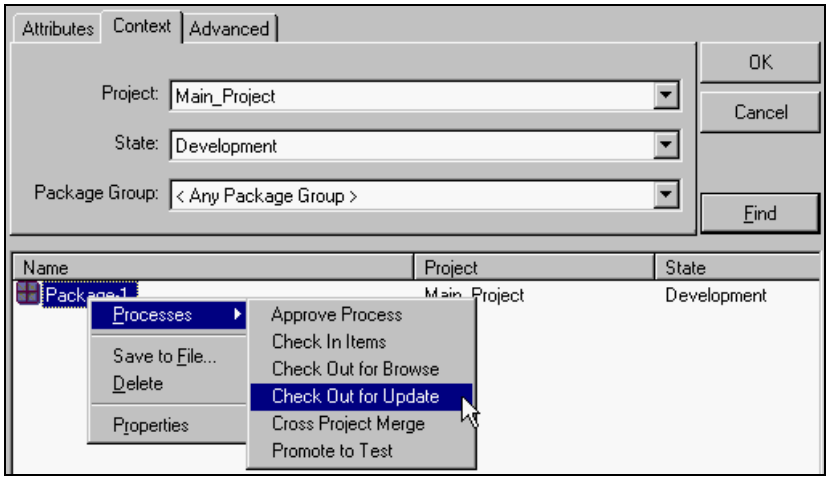
Using the Harvest package as the basis for change makes it possible to accomplish day-to-day operations seamlessly and in one operation. Version control tasks, such as checking out versions and checking in versions to Harvest, can be performed using the package as a focus of content.

Check Out

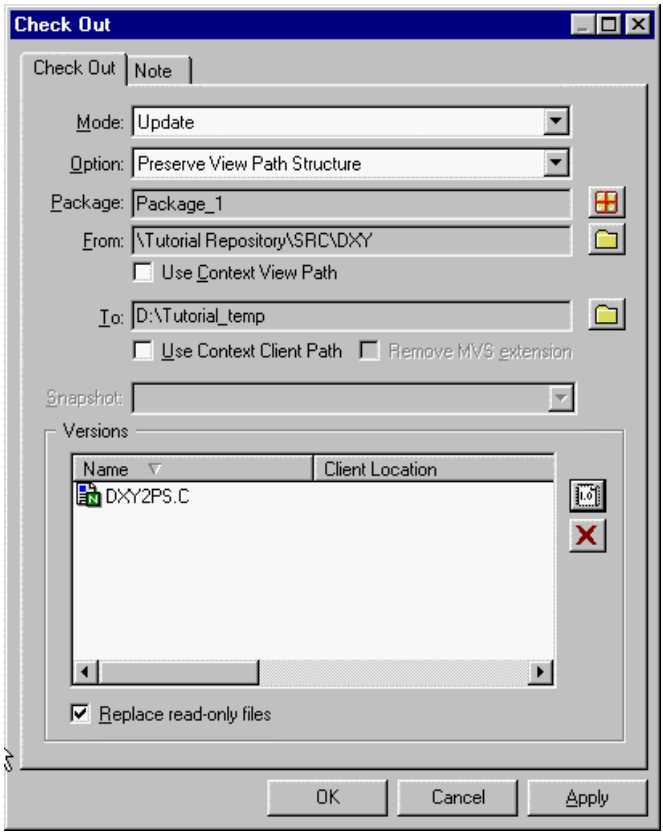
1. In the Harvest Workbench, select the Find Package utility  from the Harvest toolbar to set the user context.
2. Select the Context tab and enter the Harvest project name and state you want to check out. Click *Find.*



3. Right-click the package listed in the list box and select the *Check Out for Update* process.



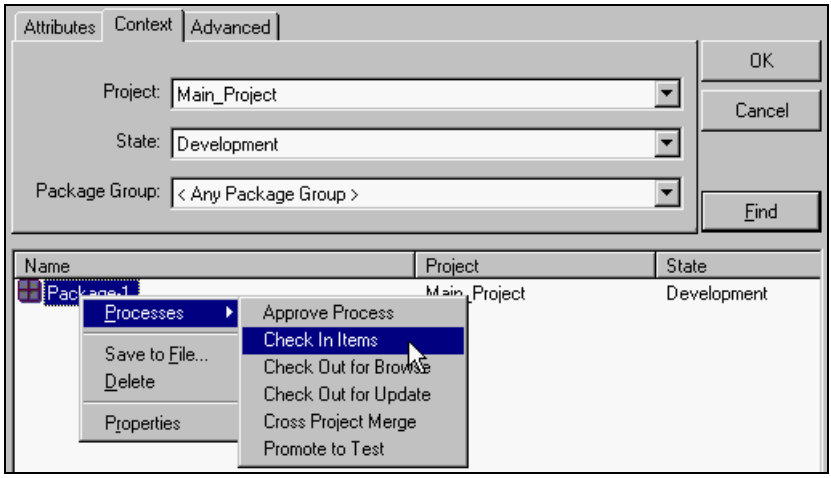
4. The Check Out dialog options, "Package," "From," and "To" will automatically pick up the user context setting. Only "Versions" need to be selected.



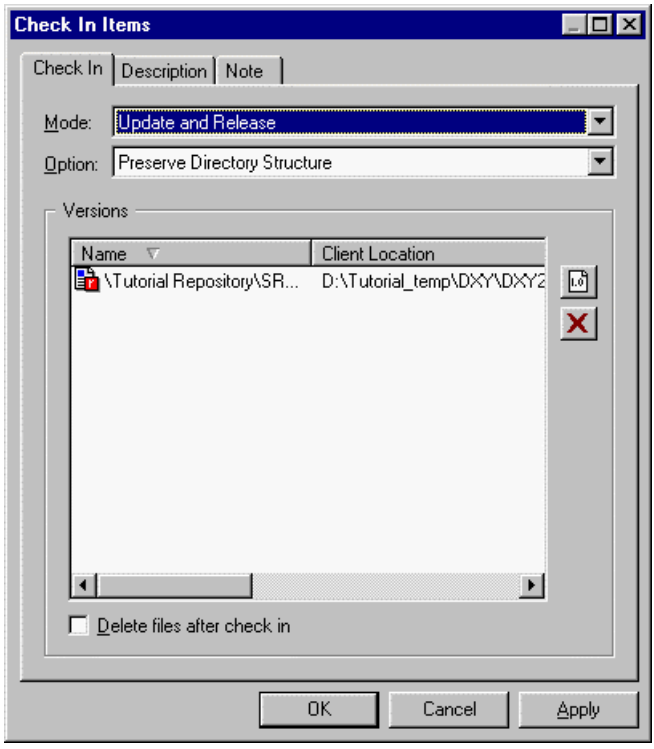
5. Select OK to execute.

Check In

1. In the Harvest Workbench, select the Find Package utility from the Harvest toolbar and search for the package the Check Out process was executed on previously.
2. The package name appears in the list box. Right-click the package name and select the *Check In Items*.



When the Check In process is executed on a package that contains reserved versions, the process automatically picks up the reserve tag versions and the corresponding client path location. No further steps are needed to complete the Check In process.



3. Select OK to check in the new versions.

What's Next?

Now you can see how the "package" based operations of Harvest can really automate many daily CM activities.

Next, you will see how these forms are associated with their particular packages, how to work with these forms, and how they are used to track the progress of your packages through their life cycles.

Chapter 5

Using Forms To Organize Pertinent Information

Track Issues During The Life Cycle Process

AllFusion Harvest forms can be used in much the same manner that paper forms are used. For example, they can be used to track issues and problems or as a structured method of communication. In Harvest, a package is the basic unit of work that moves through the life cycle and the associated form provides the link between Harvest's change management functions and problem tracking. Forms represent a way of maintaining and organizing information within Harvest. A form's modification history is maintained and can be easily reviewed.

Forms can also be associated with other packages, and therefore, with other forms, allowing information to be cross-referenced. For example, you might associate a customer contact form with a problem reported by that customer, and divide the problem fix between two packages. All the information in a form is directly accessible from the associated package.

Package And Form Association

Forms attain their greatest usefulness through association with packages, and can only be created in association with one or many packages.

There are three ways to associate a form with a package:

- One form can be associated with a package by defining the Create Package process to automatically create a form when executed. The type of form to be associated with the package is defined by the Administrator when the process is set up.
- You can associate a form with a package by using the Forms tab of the Package Properties dialog.
- You can add and associate a form with a package by right-clicking the package and choosing *Add New Form* from the shortcut menu to open the Add New Form dialog.

All the information gathered through a form is also directly accessible from the associated package. There are basically two package-to-form relationships:

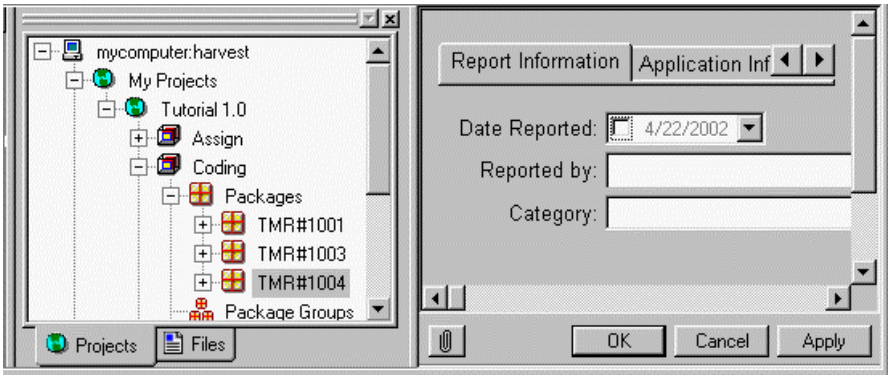
- **One Package to One Form**—This one-to-one association is the simplest relationship between forms and packages.
- **One Package to Multiple Forms**—This one-to-many association is useful to resolve several related problems. Multiple forms can be associated with it.

Harvest provides these default form types for the following organizational needs:

- **Application Change Request**—For change process within third party development projects.
- **Comment**—For any kind of information.
- **Defect Tracking**—For tracking defects in an application.
- **ESD Change Request**—For managing changes made to Server and Desktop file packages.
- **Modification Request**—For any type of software change request.
- **Problem Report—**For organizing and tracking information about a software problem.
- **Q And A—**For creating a database of information in question and answer format.
- **Testing Info**—For supporting and testing of packages.
- **User Contact—**For organizing and tracking customer and product information.

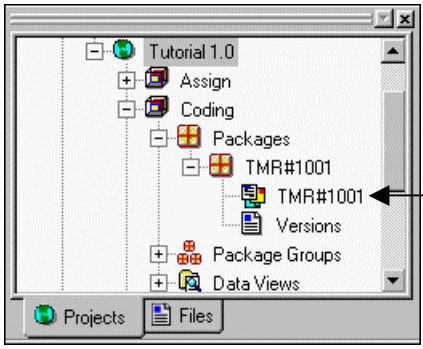
Creating A Package Associated Form

During package creation, the form name is composed of the form type and appended with the name of the package it was associated with during package creation. Harvest forms will automatically appear in the document workspace when the package is created.



NOTE: forms automatically appear in the document workspace when the package is created, if the Create Package Process was defined to automatically create a form. For information about the Create Package Process, refer to Chapter 4 in this guide.

When you expand the package, you will see its forms listed below it:



Expanding the package named "TMR#1001" will display the associated form.

NOTE: Alternatively, you may generate a full screen image of the form by pressing the F4 key on your keyboard. Pressing the ESC key will return you to the Harvest Workbench.

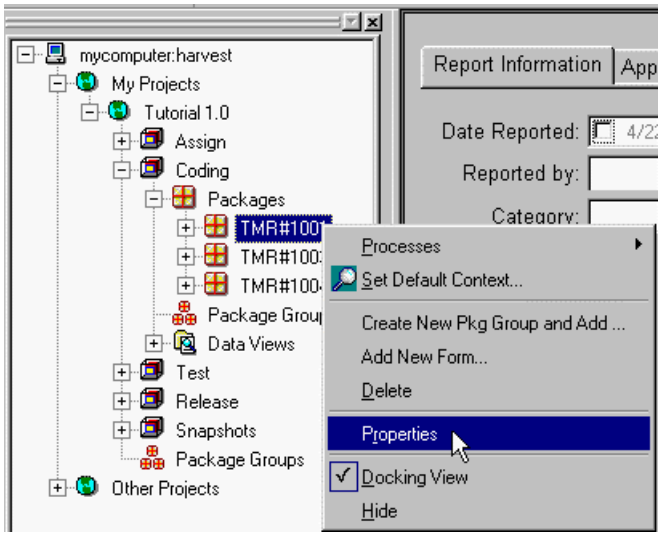
After editing the form, close it by clicking OK. Edits can be saved and you may continue to work with the form by clicking *Apply*. Clicking *Cancel* closes the form without saving any information.

Create The Association: Part One

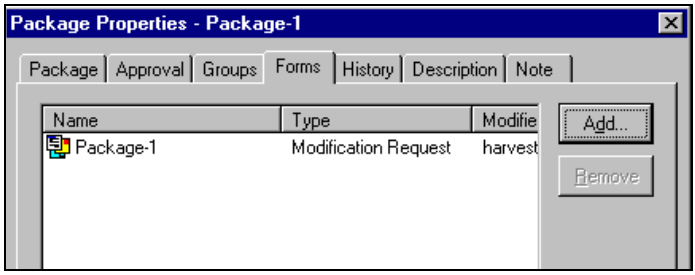
To create an association between a package and a form:

1. On the Projects tab  of the workbench, right-click the package for which you want to create the association.

2. Choose *Properties* from the shortcut menu.

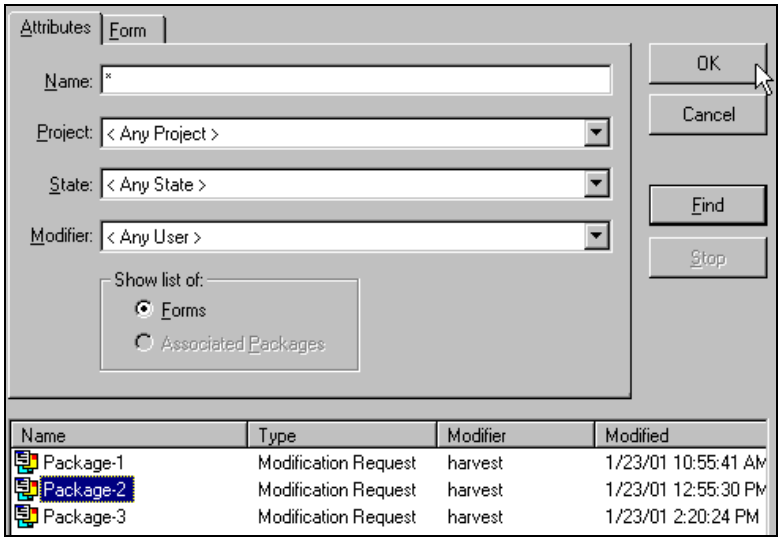


3. On the Package Properties dialog, select the Forms tab. You will see the forms already associated with the package.

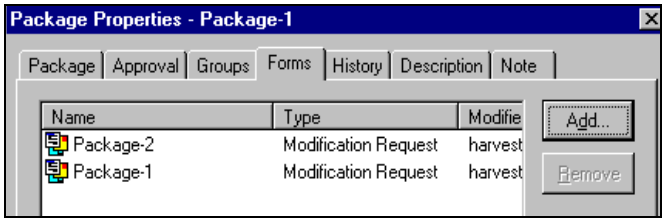


4. Click *Add* to open the Find Form dialog.

5. Using the Find Form dialog, locate and select the form you want to associate with the package and click *OK*.



6. This associates the selected form with the package. Click *OK* at the bottom of the Package Properties dialog.



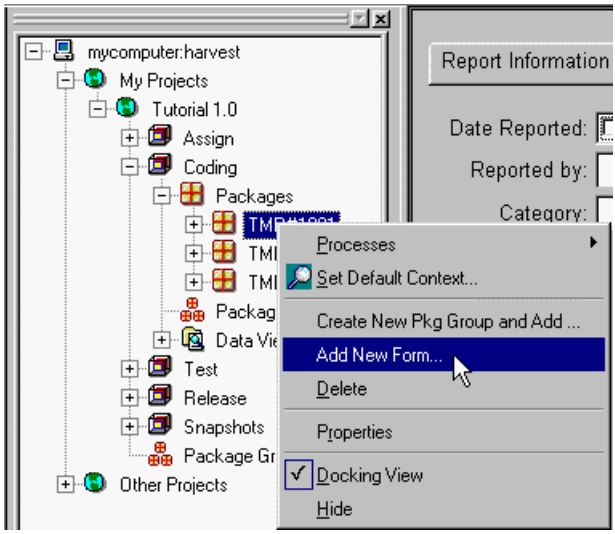
The associated forms will be listed below the package on the workbench.

Create The Association: Part Two

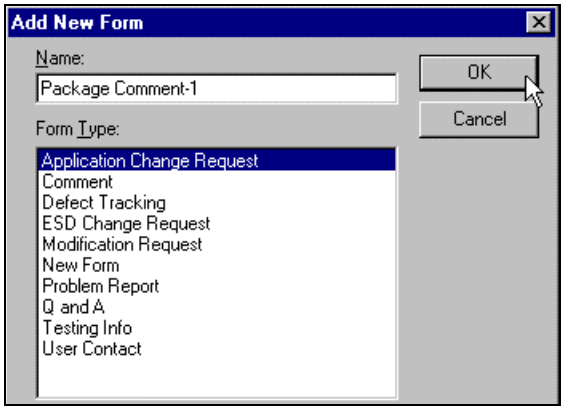
There is also another way in which to create an association between a form and a package.

1. On the Projects tab  of the workbench, right-click the package from which you want to create the association.

2. Choose *Add New Form* from the shortcut menu.

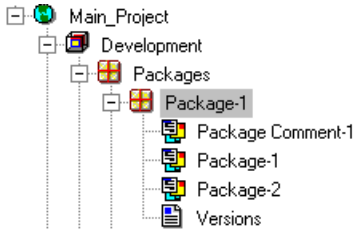


3. On the Add New Form dialog, name the form, select the type of form you want to add, and then click *OK*.



NOTE: A description of the default form types supplied by Harvest can be found at the beginning of this chapter.

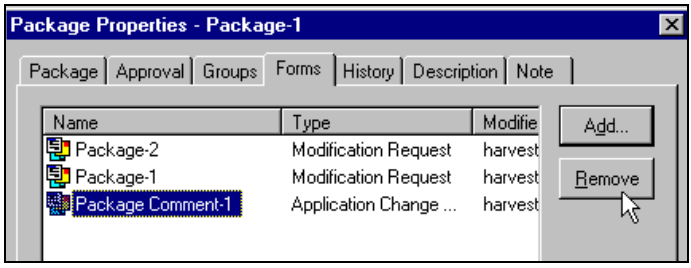
The associated form will be listed below the package on the workbench and will be displayed in the Form Viewer in the list view.



Removing An Association Between Packages And Forms

To remove an association between a package and a form:

1. On the Projects tab of the workbench, right-click the package from which you want to remove the association.
2. Choose *Properties* from the shortcut menu.
3. On the Package Properties dialog, select the Associated Forms tab.
4. Select the form you want to disassociate and click *Remove*.



5. Select *Yes* to confirm that you wish to remove the package/form association.

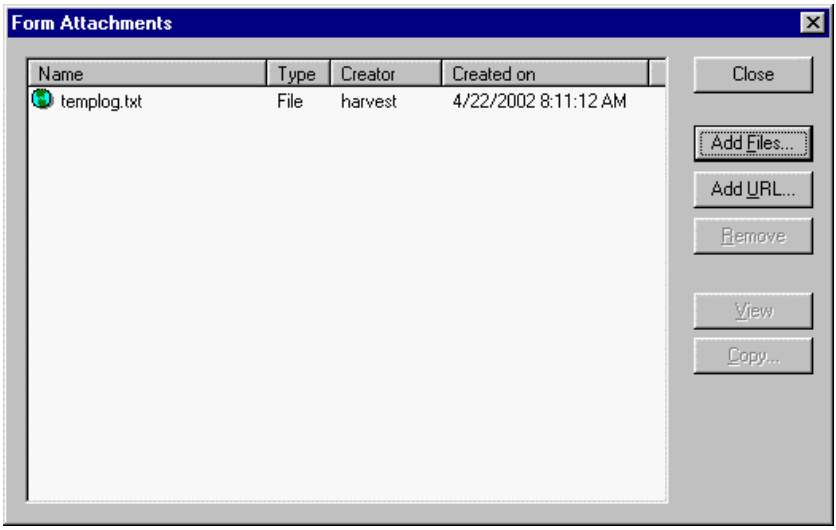
   The associated form will no longer be listed below the package on the workbench.

Using Form Attachments

Each form can have one or more attachments associated with it, linking the form to additional relevant information. You can create form attachments, and by selecting an attachment you can reference web sites, view files or copy files.

To access the Form Attachment dialog, click the paper clip icon button located on the lower left-hand corner of the Form Viewer.

The Form Attachment dialog lists existing form attachments and allows you to add, remove, view, and copy attachments.

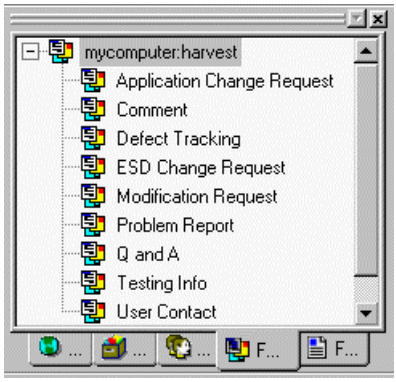


Form Wizard: Making It Easy To Create Harvest Custom Forms

Harvest makes it easier than ever to design a custom form. Rather than having to take a class in order to understand how to create a form, Harvest's Form Wizard creates it for you!

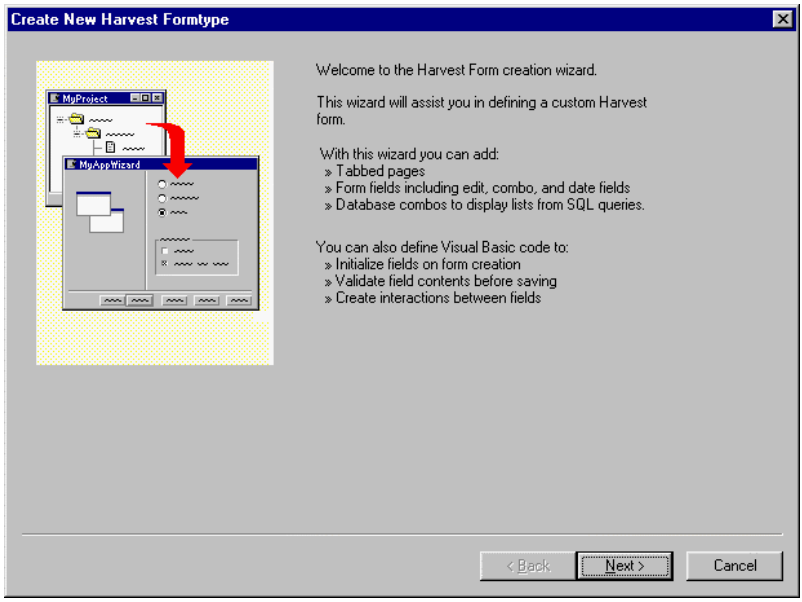
To create a Harvest form using the Form Wizard, follow these easy steps:

1. Select the Forms tab  on the Administrator's window.



2. Right-click the broker icon and choose *New Form Type* from the shortcut menu. This starts the Form Wizard.

   The Create New Harvest Form Type welcome window appears:



3. Click *Next* to continue.

NOTE: For complete details on creating a custom form and adding it to Harvest, please see the *Administrator Guide*.

What's Next?

Now that we have covered how Harvest Forms represent a way of maintaining and organizing information within Harvest, let's talk about Harvest's web interface, and how easy it is to view and configure the interface for your specific needs.

Chapter 6

Using The Web Interface

Easy Access With A Web Browser

The Harvest web interface (referred to simply as Harweb) offers Harvest users access to the CCM system, using a web browser. Harweb supports both Harvest administrative and user functions. The functions available are identical to the Harvest graphical user interface.

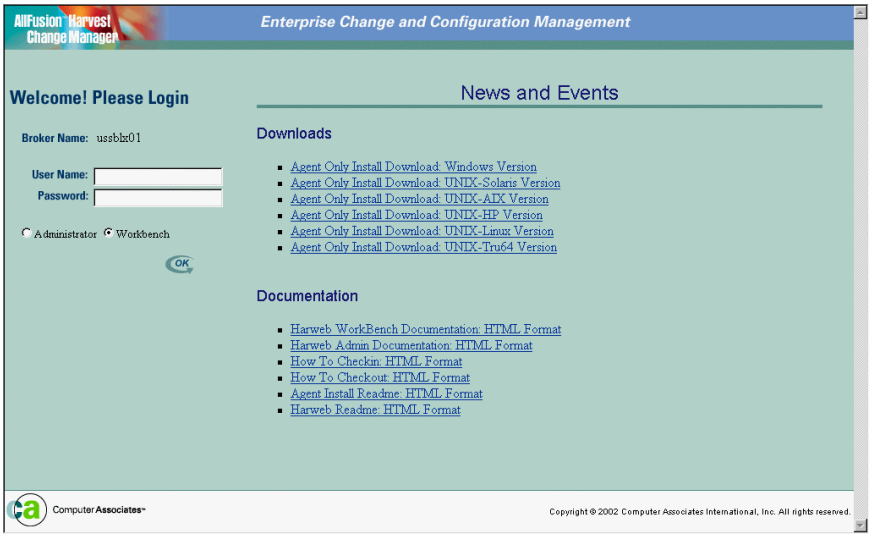
Logging In

Whether you are logging in as an Administrator or as a User, follow the steps below:

1. Invoke the web browser and in the address field enter the location of the Harweb Home, for example:

   http://*web_server_name*/harweb

2. The login page allows the choice of accessing the Harvest Administrator or Workbench.



NOTE: You can open multiple instances of the Harvest interface by logging into Harweb, and thereafter, typing Ctrl + N to open additional instances.

Using the Administrator Interface

The administrative web interface offers functions similar to the Harvest graphical user interface (GUI) Administrator window.

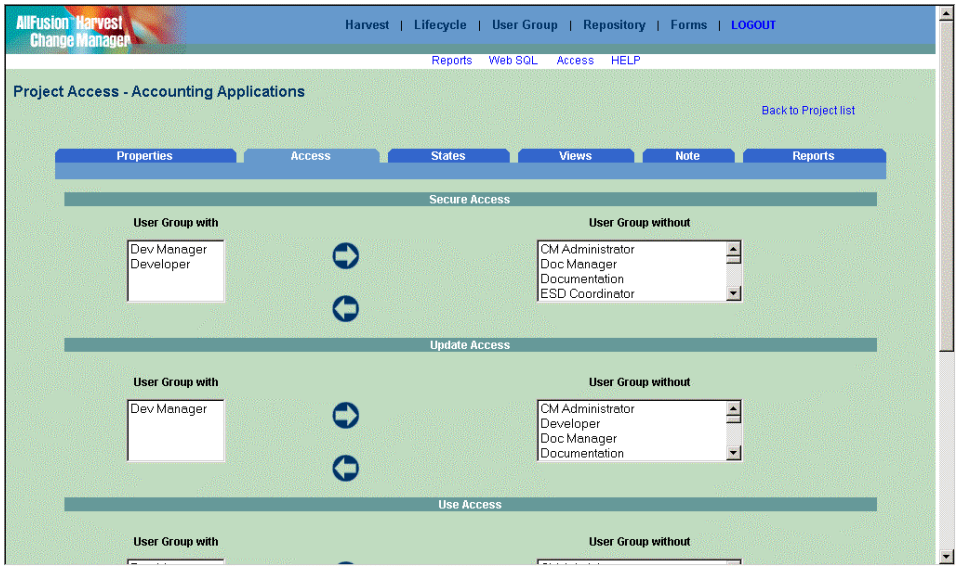
Menus across the top of the page allow you to access Harvest set access, life cycle, user group and user, repository and logout functions.

- From the Harvest menu you set Harvest access control.
- From the Lifecycle menu you can list active, inactive and template projects, and create projects.
- From the User Groups menu you can list users and user groups, and create users and user groups.
- From the Repositories menu you can list repositories and create repositories.
- From the Forms menu you can generate custom forms.

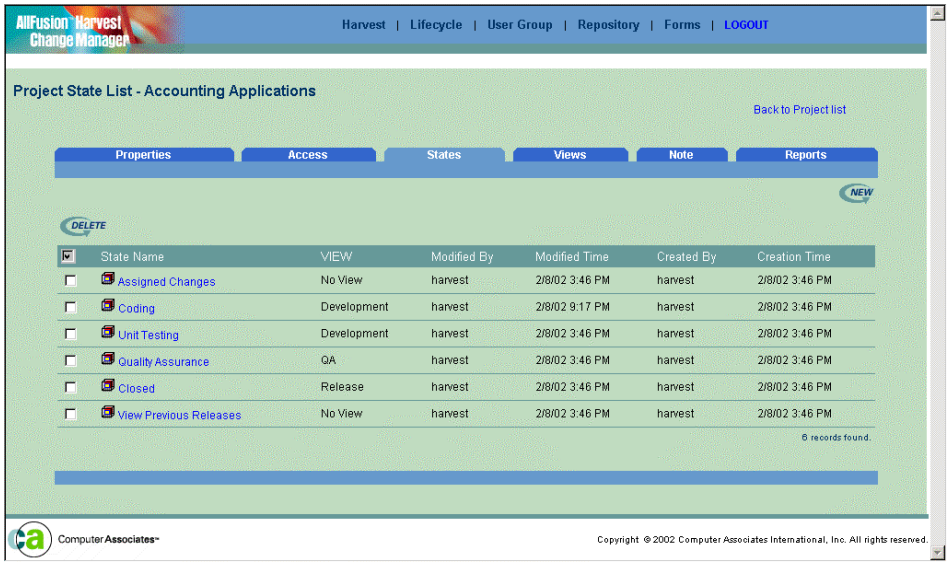
Active Project List

The Active Project List page shows all projects that are designated as active. Projects are listed by name with columns displaying their attributes.

The Project Properties page displays when you click a project name listed in the Project Name column of the Active Project List. On the Project Properties page, six tab pages allow you to view and modify the project properties.



The Project State List page shows all the states belonging to your current project. States are listed by their sequence in the project, with columns displaying their attributes. The State Properties page displays when you click a state name.



The State Process List page shows all the processes belonging to your current state. Processes are listed by their sequence in the state, with columns displaying their attributes. You can add, delete processes using the State Process List page.

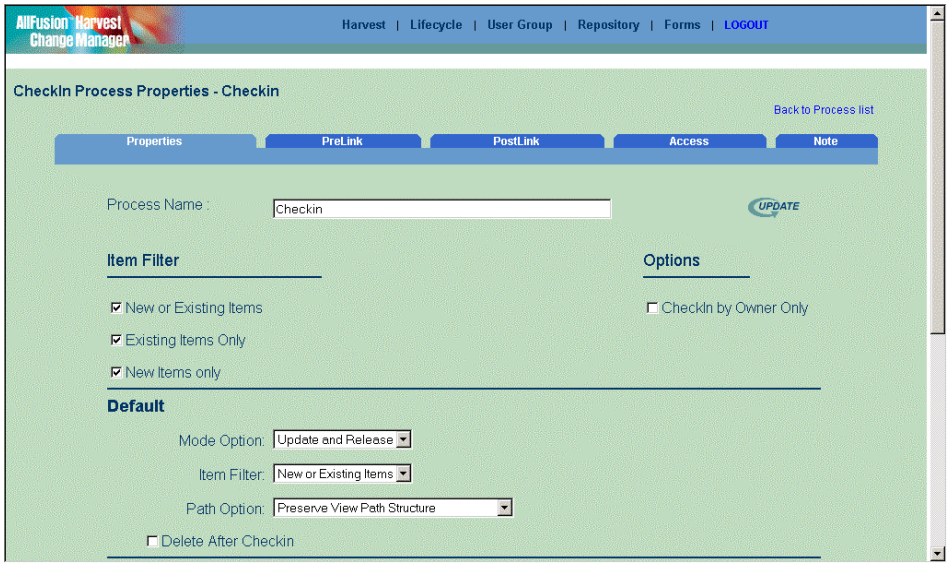
The Process Properties page displays when you click a process name.

Define Check In Process

The Check In Properties page allows you to specify options enabled for the process, and specifies defaults for various execution options.

The Check In process allows users to copy files located in a directory on the client into the current Harvest context. If a corresponding item already exists in the view, it must be reserved by the package specified during check in.

Linked processes execute before and after the check in process completes successfully. In this case, successfully means that at least one file is checked in.



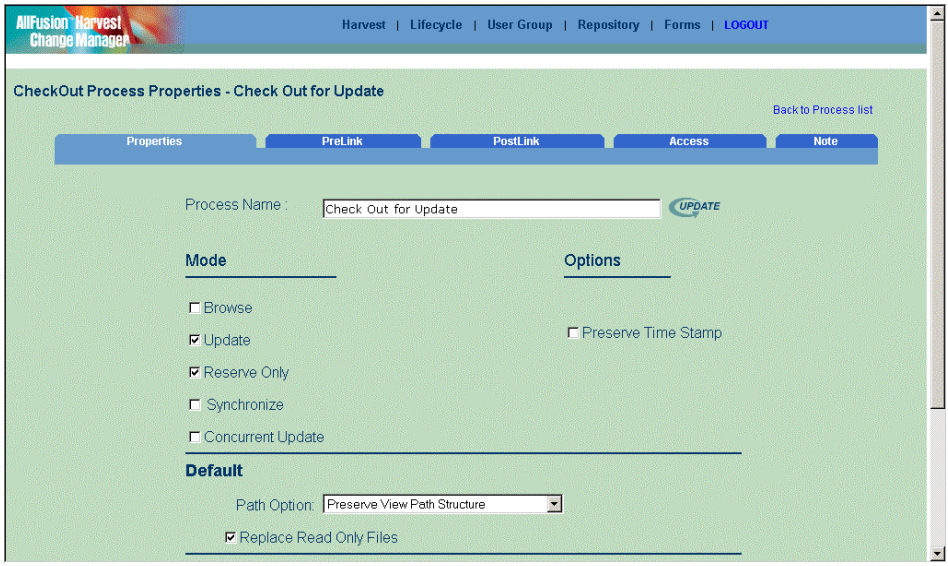
When a check in process is defined, you can decide which item filters to use. The available item filters are:

- **New or Existing Items**. All selected files are checked in if they are reserved by the package or did not previously exist in the repository.
- **Existing Items Only**. This option limits the check in to files that have corresponding items reserved by the package.
- **New Items Only**. This option limits the check in to files that do not have corresponding items in the repository.

Define Check Out Process

The Check Out Properties page allows you to select the check out mode or modes to be allowed, and specify default values for various execution options.

Linked processes execute before and after the check out process completes successfully. In this case, successfully means that at least one item is checked out.



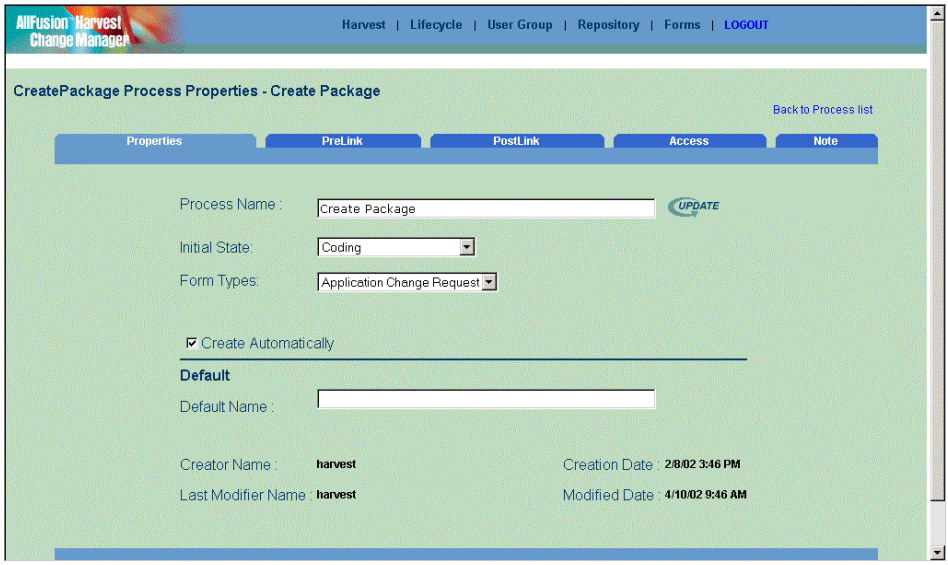
The check out process supports five different modes of operation:

- **Browse**—checks out the items to the destination directory but does not reserve the item and does not allow the user to check the files back in. The read-only attribute is set on files checked out in Browse mode.
- **Update**—copies selected versions to the destination client directory and creates a reserved version on each item's trunk, allowing the corresponding files to be checked back in. The permission on a read-only file is changed to normal (write access) when this mode of check out is used.
- **Reserve only**—does not move any data to external directories but creates a reserved version on each item's trunk so that corresponding files can be checked in.
- **Synchronize**—compares the versions selected for check out with the version information maintained in the signature file for the destination directory. Versions that do not match the version on the file system (newer or older) are checked out in Browse mode. For more information on signature files, see the *User Guide*.
- **Concurrent update**—copies selected versions to the destination client directory and creates a reserved version on a package branch for each item. All package updates accumulate on this branch. The permission on a read-only file is changed to normal (write access) when this mode of check out is used.

Define Create Package Process

The Create Package Properties page allows you to specify whether a form should be created and automatically associated with the package created by this process, and if so, what kind of form. Finally, you can specify an initial state for the package to be created in and a default name for packages created with this process.

Notify and UDP processes can be linked to a create package process using the Pre-Linked or the Post-Linked process. Linked processes execute after the OK or Apply button is clicked on a package/form combination.

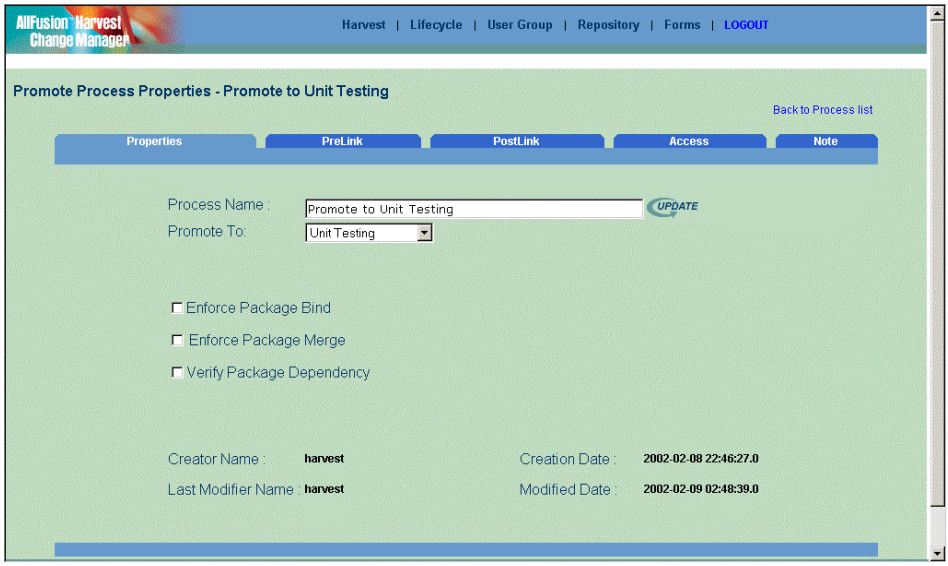


Define Promote Process

The Promote Properties page allows you to specify the state to which packages should move when this process is executed.

The design of other life cycles might require that all package branches be merged before a package can advance in the life cycle. The Promote Properties page allows the user the option of allowing unmerged packages to be promoted.

Processes pre-linked to a promote process are executed after packages are verified for promotion. If a package cannot be promoted, no linked processes are executed. The state of the packages is changed only after the successful completion of all pre-linked processes defined in the pre-linked process. Post-linked processes execute after the package has been successfully promoted.



- **Enforce Package Bind**—Enabling this option causes all the packages belonging to a bound package group to be promoted together.
- **Enforce Package Merge**—Enabling this option prohibits packages to be promoted to the next state if they are merge-tagged, reserved-tagged or associated with branch versions. If you enforce that packages must be merged, two requirements must be met:
    1. The latest change for the package must be on the trunk, not on a branch.
    2. The latest package version must not be tagged as merged (M) or reserved (R). The interactive merge process must be used to resolve the merge tag before the package can be promoted.
- **Verify Package Dependency**—Enabling this option disallows packages to be promoted to the destination state until all dependent packages are located in the current state.

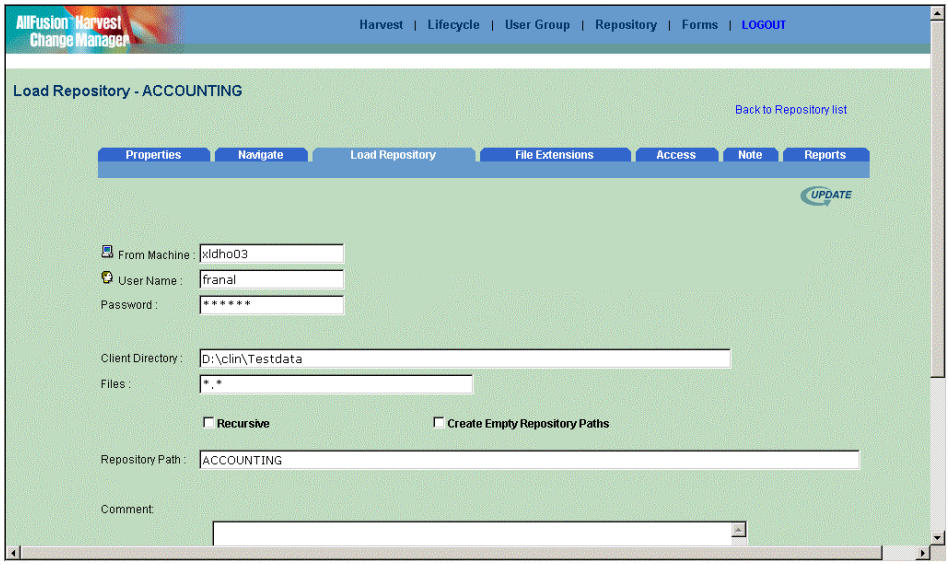
Repository Properties

The Repository Properties page displays when you click a repository name listed in the Repository Name column of the Repository List. On the Repository Properties page, seven tab pages allow you to view and modify the repository properties.

Tip: When you create a repository, you cannot continue with the next step of loading items until you save the name and directory information. To do this, click Update to save the specified information.

Load Repository

After a repository is initially created with the Repository Properties page, the Load Repository operation brings files located in operating system directories into the Harvest repository.

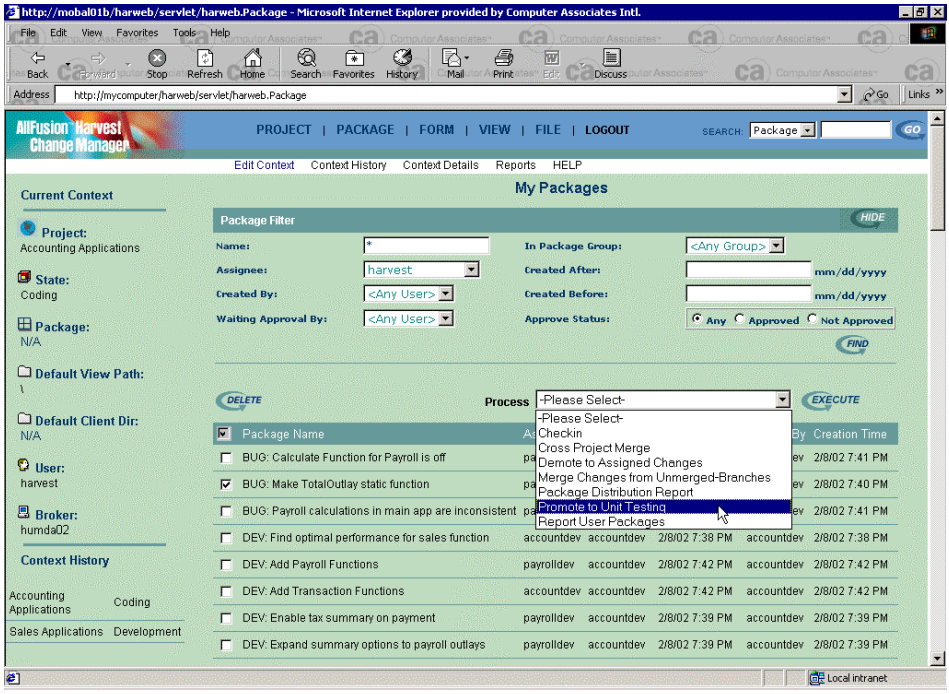


Using The Harweb Workbench

Using Harvest's web interface is easy! The interface functions similarly to the Harvest Workbench, therefore, you will be able to find and work with your projects effortlessly. You select a context, select a package or version, and then execute processes defined for the context. An assortment of options is available from menus on each page of the interface.

Tip: A Harvest context consists of a project, state, view, package, and item path.

The following is a picture of Harvests web interface, after you have logged in, showing the My Packages page:



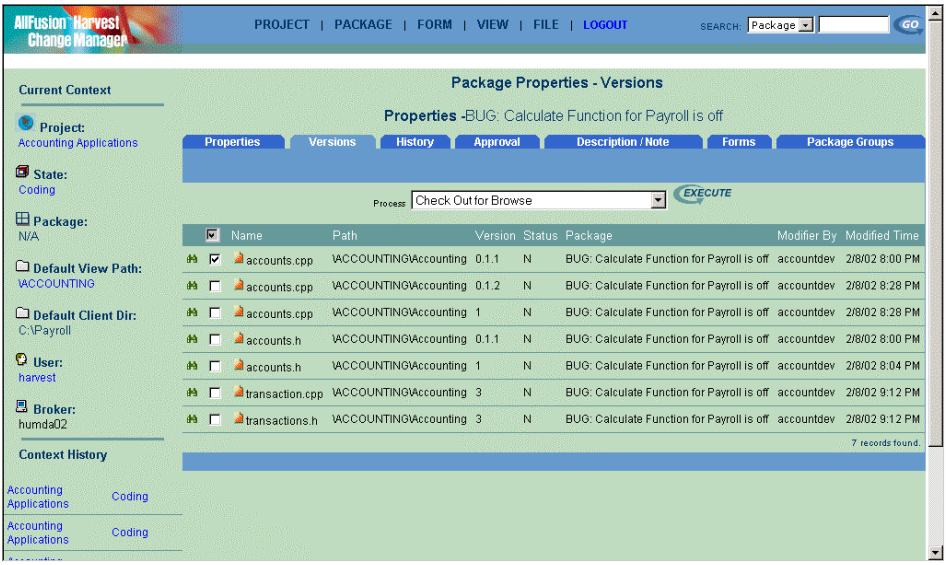
My Packages

The My Packages page (displayed after logging in or from the Package menu), lists all packages assigned to you for action in the current project or state, thus enabling you to focus on specific tasks. Packages are listed by name.

Tip: Your current context is shown on the left-hand side of the page, and you can open the Properties page for a Harvest project, state, package, or user, by clicking the name.

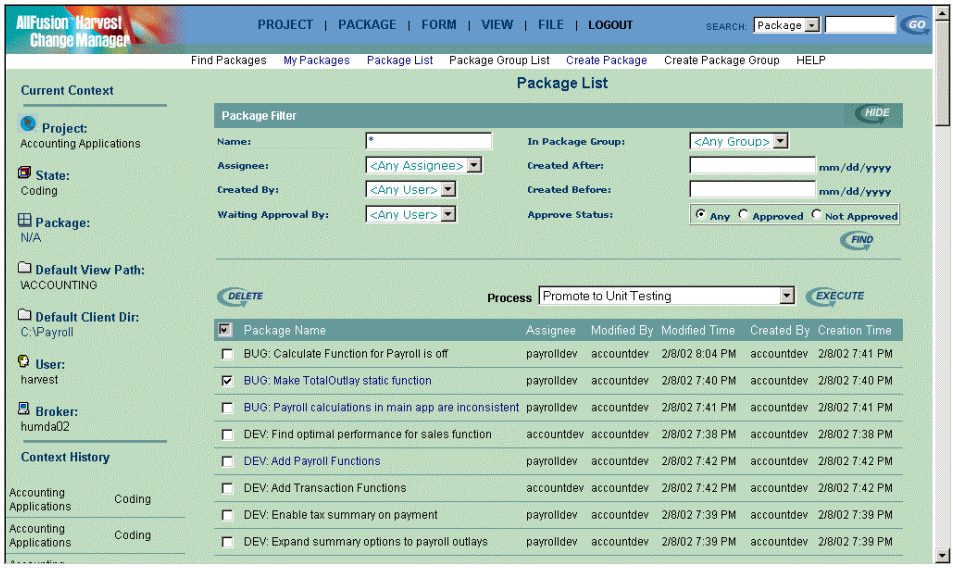
Package Properties

Clicking a package name on a package list opens the Package Properties page. Using the Package Properties page, you can view and define package attributes, including opening associated forms and associating the package to one or more package groups.



Executing Processes

The execution of each Harweb process is performed from a package, item, or version list. A package, item, or version must be selected and your Harvest context set, before you can successfully execute a Harvest process. Choosing a process from the Process drop-down list and clicking the Execute button, opens a process execution page that allows you to select different options.

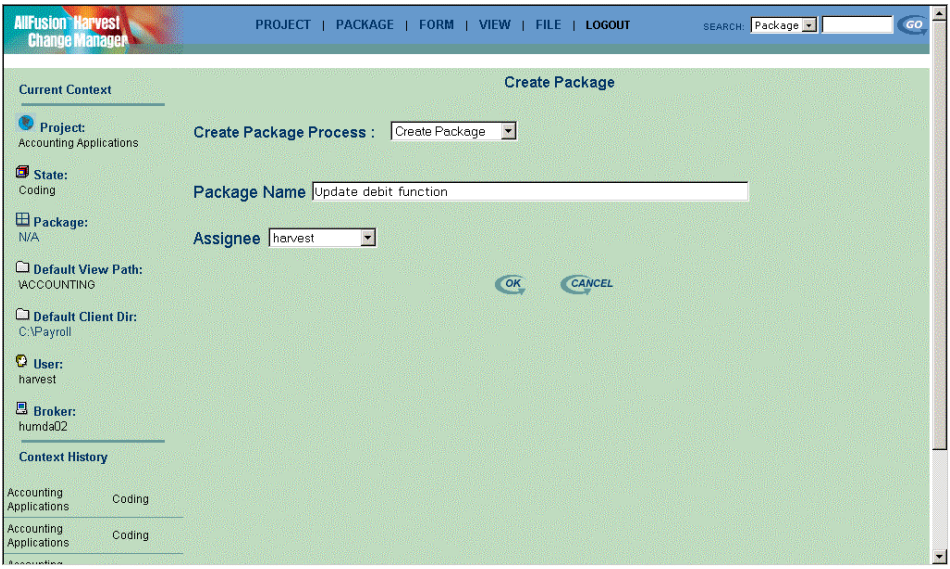


Tip: You can easily approve a package by checking the box next to the package you wish to approve, selecting *Approve* from the drop-down list, and the clicking *Execute*.

The results of the execution are shown in the Process Log page. You can view details about the process execution by clicking the Show/Hide button.

Creating A Package

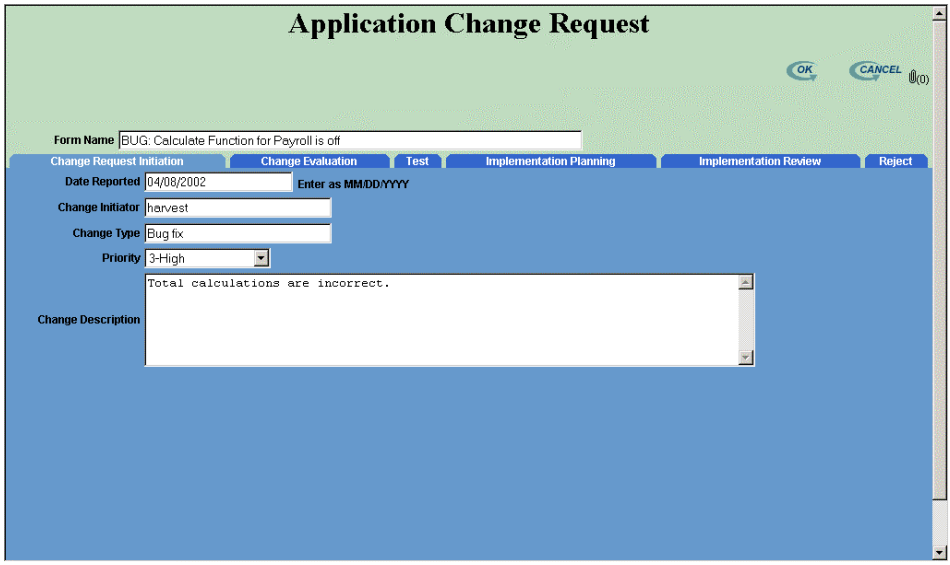
Choosing the Create Package option from the Package menu opens a page that allows you to create a package in the state defined as the initial state in its process properties dialog. In addition, if the form option *Create automatically* was selected in the process properties dialog, a form with the same name is created as the package is created and the two objects are automatically associated.



After creating a package, it is listed in the Package List. Clicking the package name opens its Package Properties page.

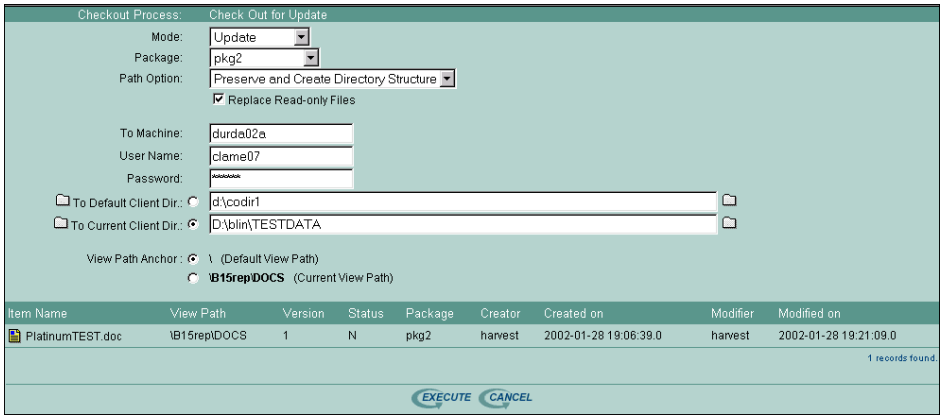
Editing A Form

Choosing *Form*, the Forms list shows a list of the forms associated with your current context. Forms are displayed and can be modified when you click a form name to open the form.



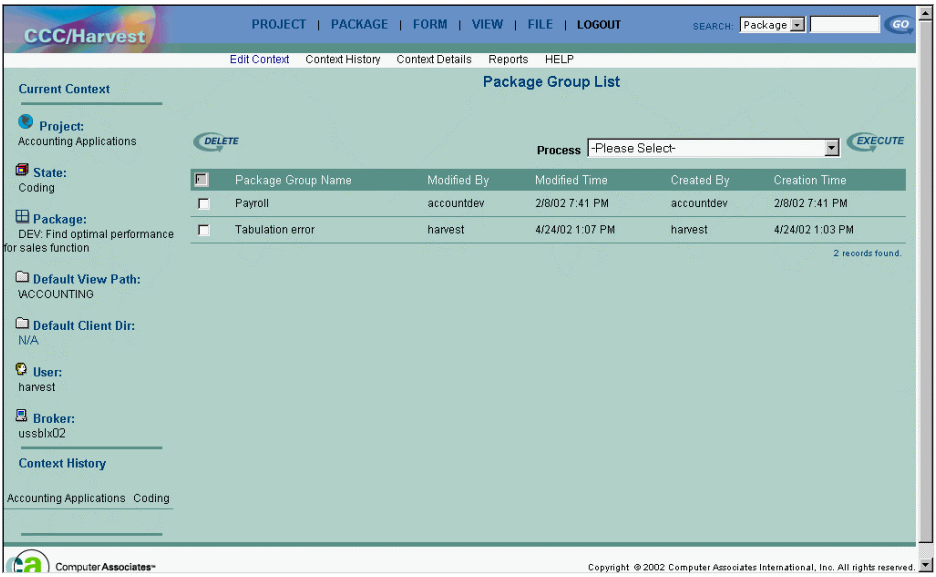
Using The Check In and Check Out Processes

Using the Harweb interface, you can check in files, packages with versions, and files from remote locations. You can also check out items, packages with versions, and items to remote locations. The check in and check out pages vary according to the type of check in or check out you are performing.



Promoting A Package Group

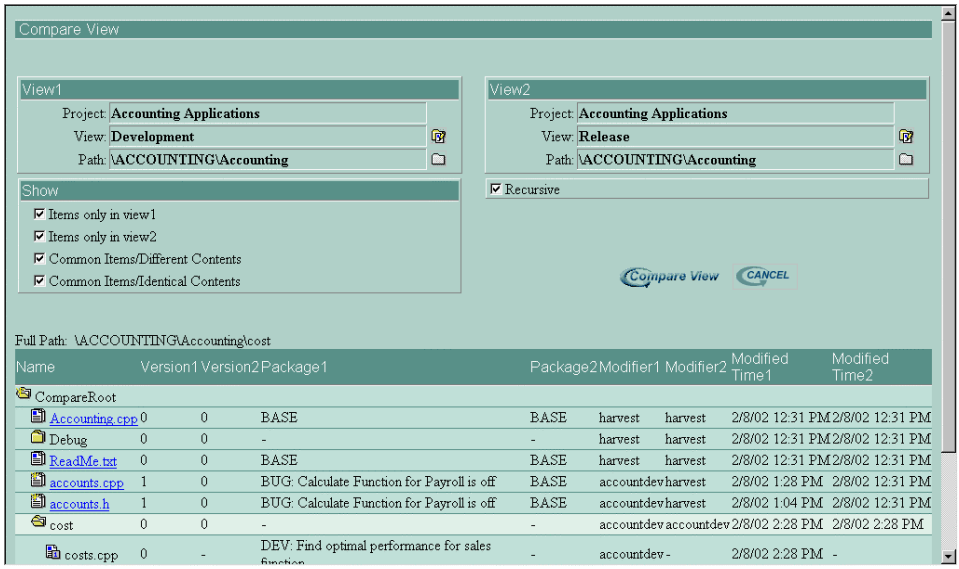
Choosing the Package Group List option from the Package menu opens a page that lists all the package groups in your current project or state, and allows you to execute a promote process on a package group.



Comparing Views

The compare views process generates a report showing the differences between any two views, either snapshot or working, which exist in any project. You can select two views to compare and a variety of options to generate the report.

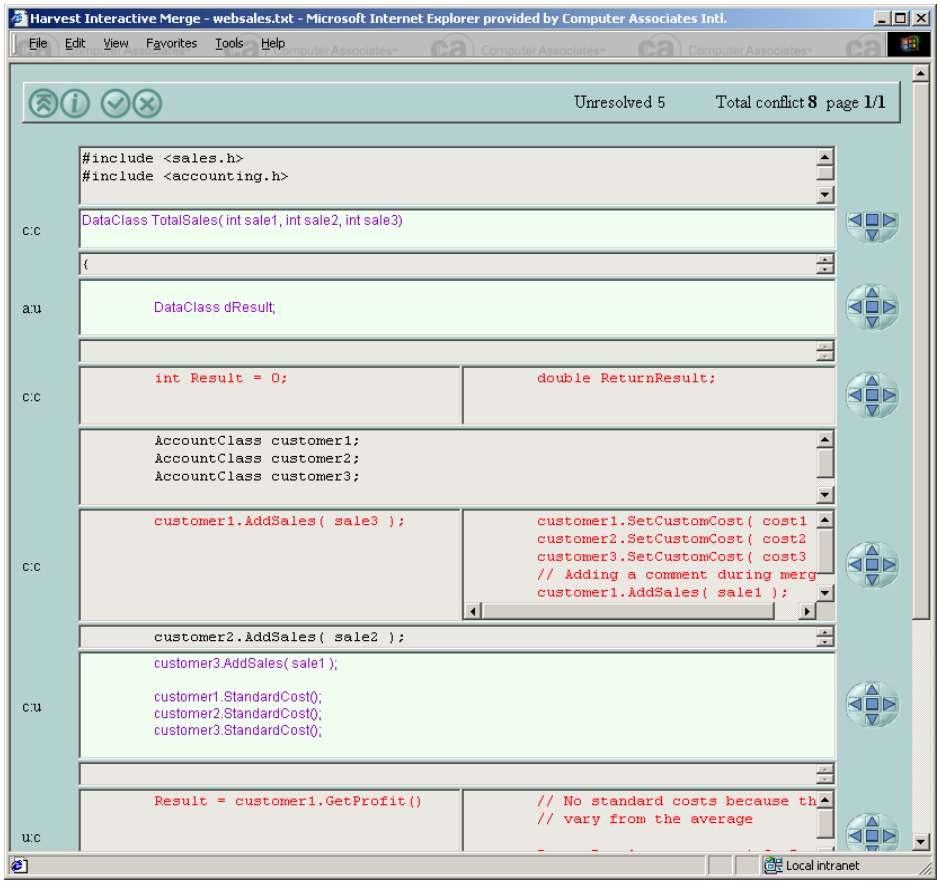
Once View1 and View2 are selected and you have specified your item selection, click the Compare View button to show a list of items meeting your criteria.



Merging Files Interactively

The interactive merge process enables you to resolve conflicts and to combine changes between two versions of a file or to resolve conflicts in a merge-tagged version after it is created by the concurrent merge or cross project merge process.

To resolve a merge tag, select the merge-tagged version from a version list before invoking the interactive merge process from the Processes menu. You can select merge tags from the following version lists: Package Properties Versions page, Item Versions page, and Find Version Versions list.

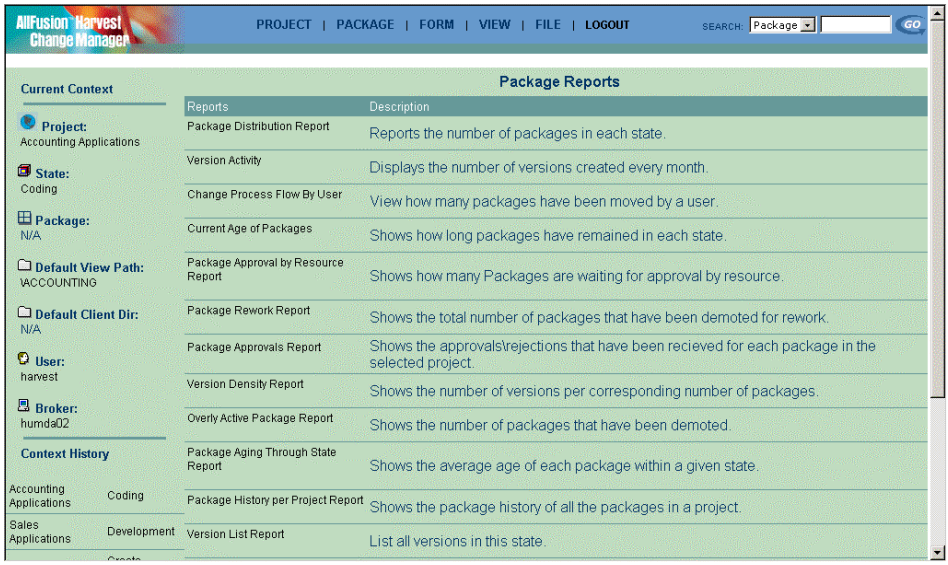


The Interactive Merge page is a continuous page representing a page of the merged document. The two versions of the item being merged display with common blocks and conflict blocks. Conflict blocks appear side-by-side, and you must select one block in favor of the other using the take button. Common blocks extend across the width of the page. You can use the scroll bar on the right side of the page to move up and down in the file.

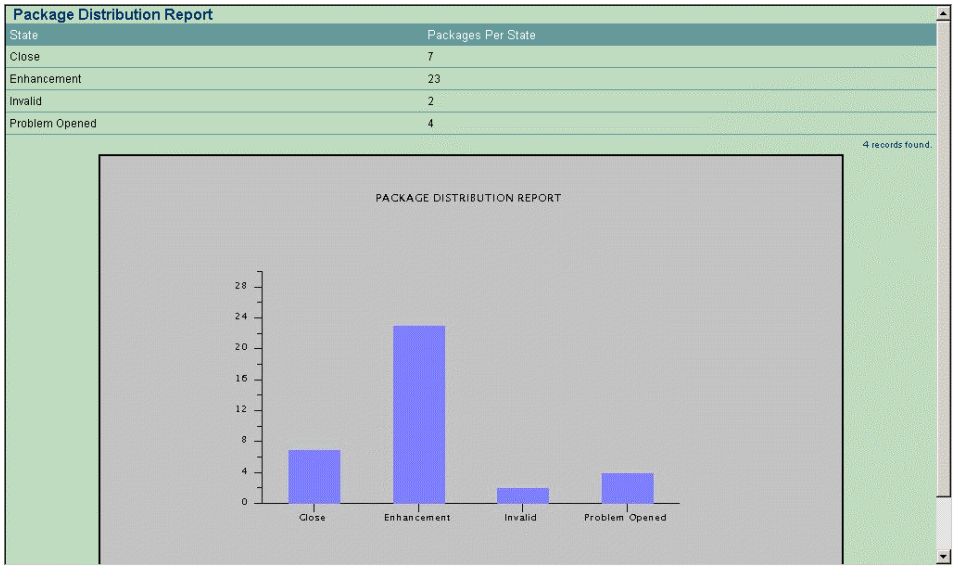
When the merge is completed, the merge-tagged version is replaced by a new, untagged version.

Reporting

Harweb reports incorporate package management information from Harvest. Using the Package Reports page, you are able to generate package-related reports specific to your current Harvest context. A variety of reports are listed, and you can click a report name to select and generate a report.



Shown below, is the Package Distribution report, used to view statistics regarding the total number of packages within each state. The report estimates the number and distribution of packages that need to be completed.



Chapter 7

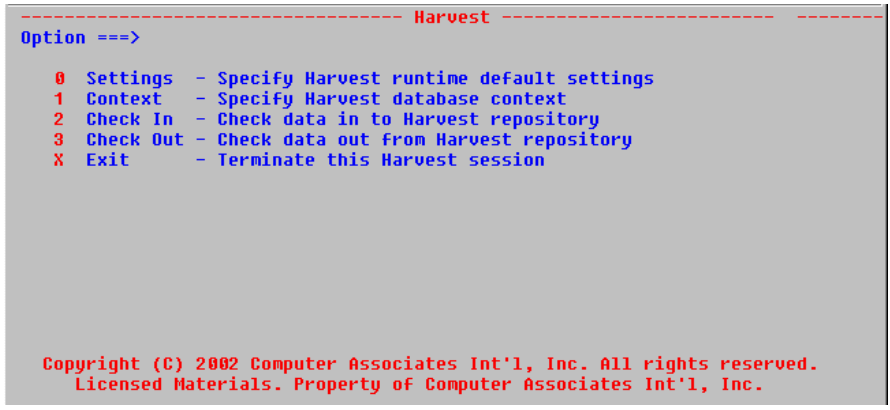
OS/390 Users—Integrate Development

Availability For The ISPF Client Interface

Harvest is available as an ISPF in a standard NONGUI menu driven interface to the OS/390 operating system and file utilities. The Harvest ISPF interface acts as an extension to this environment. It offers a subset of functions that enables OS/390 users to check data into and out of the Harvest database without requiring them to leave their ISPF environment.

Harvest ISPF Primary Panel

When you invoke the ISPF Harvest interface, the Login panel displays. After a successful login, the Harvest ISPF Primary Panel displays. The primary panel is shown below:



The primary panel offers you selections for settings, context, Check In and Check Out. Selecting Option `0`, Settings, allows you to set Harvest ISPF usage default values, which remain in effect between sessions, until they are changed.

Harvest Context

The context consists of the Harvest project, state, view, package and item path. A valid context must be set prior to executing a check in or check out. You set these values once, and they remain in effect until they are changed. Selecting option `1` displays a window that allows you to set the context and map ISPF processes to a Harvest process defined for that context.

The Check In Panel

The Check In panel allows you to copy files from the OS/390 data sets into Harvest, creating new items or updating existing items that have changed.

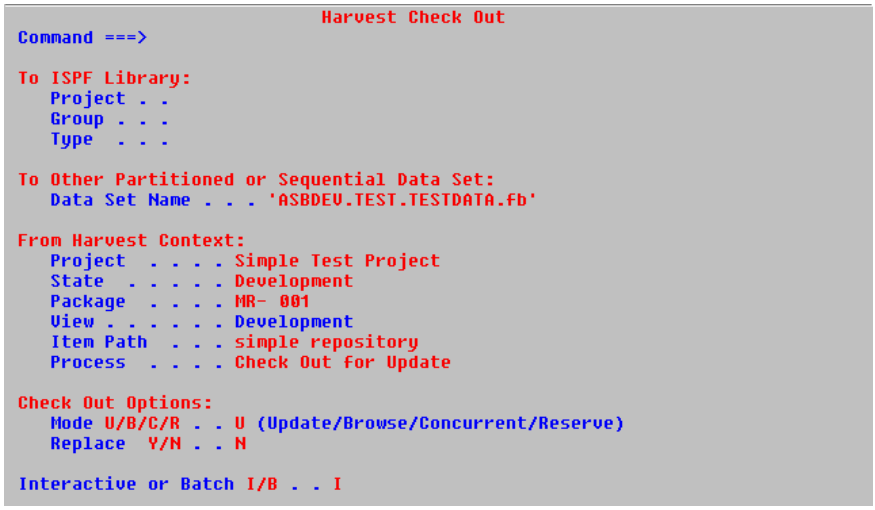
To check in an existing item, it must be reserved by a package. If the item was checked out for concurrent update, the Check In process will update the current package's branch. If the item was checked out for update or reserve only, the following screen appears:

```
                          Harvest Check In
Command ===>

From ISPF Library:
   Project . . .
   Group . . . .
   Type  . . . .
   Member  . . .             (Pattern for list, blank or "*" for all members)

From Other Partitioned or Sequential Data Set:
   Data Set Name . . . 'ASBDEV.TEST.TESTDATA.fb.cpp'

To Harvest Context:
   Project  . . . Simple Test Project
   State  . . . . Development
   Package  . . . MR- 001
   View . . . . . Development
   Item Path  . . simple repository
   Process  . . . Check In Items
Options:
   Mode UR/RO/UK . . UR  Path  P/PC/A . . A   Filter NE/N/E . . NE
   Delete    Y/N . . N   Comments Y/N . . N
Interactive or Batch I/B . . I
```

NOTE: For a complete description of the Check In process, refer to the *User Guide*.

The Check Out Panel

The Check Out panel allows you to copy versions of items under Harvest's control to OS/390 data sets where the items can be updated or browsed. Alternately, items can simply be reserved without actually copying the data to an OS/390 data set.



NOTE: For a complete description of the Check Out process, see the *User Guide*.

Chapter 8

Frequently Asked Questions

Questions And Answers

Question: What are the key differentiators that Harvest provides developers?

Answer: There are four main differentiators:

Process Driven Change and Configuration Control. The ability to provide an automated repeatable process for application development and to extend this process automation through the build cycle as well. This ensures that development for a specific application can be performed the same way every time and that once changes are made, the application can be built or re-built the exact same way every time.

Flexible and easily customizable life cycle and process models. Using the 10 standard "out-of-the-box" process models as templates, users can quickly implement CM processes and then easily customize those models to mirror their own development processes.

Enterprise scalability. Harvest's n-tier architecture supports not only the ability to scale up for large organizations, but also allows source code from all managed platforms to be under the control of a single CM repository. This eliminates the need to have multiple redundant CM servers to support team development or mixed development environments. A single Harvest server is designed to support users across an entire organization, including multiple sites.

Integrated Change and Configuration Control facilities. Harvest provides an integrated tool set for defect (problem) tracking, change management, source code control, workflow control and dependency and build management.

Question: How do you define "ease of use"? How do you deliver it?

Answer: Ease of use is targeted at three main concerns—minimizing the impact of the tool on the developer's everyday tasks; reducing training requirement and facilitating rapid implementation and easy customization. The Harvest Workbench user interface is designed to minimize the impact on training requirements in order to begin using the tool. The Harvest Workbench user interface presents a tree view in which to navigate the application life cycle; each state in the life cycle and the contents of change packages as they move through the development process. Harvest also integrates directly into standard development environments such as the Visual Developer's Studio, so that the impact on the developer's activities is minimal. Harvest delivers several sample "out-of-the-box" process models as templates that can be used to quickly implement CM processes and then easily customized to mirror the development processes for any given business or application.

Question: How does Harvest support Software Configuration Management process control?

Answer: Software Configuration Management is the process of identifying, organizing and managing modifications to applications throughout the software development life cycle. Harvest delivers several sample "out-of-the-box" process models as templates for various types of application development. These process models can be used as is to quickly implement standard CM processes and then easily customized to mirror the development processes for any given business or application. Each life cycle supports an unlimited number of automated processes for each state or stage, such as check in/out, notify, promote, approve, update/view forms, as well as user-defined processes such as the execution of external commands and scripts.

Question: How important is process-driven SCM?

Answer: Process is vitally important to application development. Most organizations comment that the physical control of software assets is just the tip of the iceberg, and that gaining control over this chaos we call application development, can only be managed through the establishment of a repeatable process for that control.

Question: What types of components, besides source code, ca Harvest manage?

Answer: Harvest is designed to manage all the artifacts required to document, manage and implement change in a single change package. A change package may typically contain MS-Word documents as design specs, online forms for problem and change tracking, log files, executables, bitmaps and test data, as well as source code.

Question: How is distributed development supported?

Answer: Harvest was designed to provide a centralized storage mechanism for distributed and remote application development. Its n-tier architecture ensures rapid access and a small footprint on the Client. In addition, for widely dispersed "follow-the-sun" application development where reliable network access is not always ensured, Harvest supports a multi-site concurrent development facility. Using this facility, Harvest supports concurrent development across the multiple distributed CM repositories.

Question: When upgrading to Harvest from an earlier release, is there any repository conversion necessary or, is the existing repository upward compatible?

Answer: A conversion utility is provided with Harvest to assist you in converting to the right format for both data and forms.

Question: Can I use Oracle as the ODBC Repository?

Answer: Yes. With Harvest, you have the ability to use Oracle as the repository.

Question: What platforms does Harvest run on?

Answer: **Server:** Windows NT, Windows 2000, SUN, Linux, AIX and HP are supported. **Client:** Windows 98, NT, 2000, SUN, AIX and HP are supported. A web browser and server are also supplied for other platforms.

Question: What are the available platforms that have the agent technology?

Answer: Windows NT/2000, Win98, Linux, HP-UX, Solaris, AIX, Compaq Tru64 UNIX, and OS/390.

Appendix A

Glossary

A Quick Reference Guide For Harvest

Process Types And Their Actions

Approve
Allows electronic sign-off of a package or package group before it can be promoted.

Check In
Creates a new version of an item by bringing the changes made to a file into Harvest.

Check Out
Copies a version of a Harvest item to an external directory and optionally reserves it.

Compare Views
A summary report that lists the difference between any two views.

Concurrent Merge
The first step of merging branch versions. Compares a selected branch version with the latest trunk version to determine conflicts.

Create Package
Creates a package and, optionally, a form with the same name, and automatically associates them.

Cross Project Merge
Merges versions between Harvest projects.

Delete Version
Deletes the latest version(s) of an item.

Demote
Returns a package and its associated versions to a previous state.

Interactive Merge
Allows users to resolve conflicts between branch and trunk versions.

List Versions
A report that lists information about the versions of items in the Harvest repository.

Move Package
Moves a package from a state in one project to a state in another project.

Notify
Sends an email message to selected Users or User Groups.

Promote
Moves a package and its associated versions to another state further in the life cycle.

Remove Item
Logically deletes an item from the repository.

Take Snapshot
Takes a read-only image of a working view at a certain point in time.

User-Defined Process (UDP)
Executes a user-defined process.

Other Common Terms Used In Harvest

Forms
Harvest forms can be used in much like paper forms are used. For example, they can be used to track issues and problems or as a structured method of communication. Forms exist at the global level with Harvest, which means they do not belong to a particular project, package, or state, but can be used by any. Forms attain their greatest usefulness through association with packages. This association provides the link between the problem tracing and change management functions in Harvest. All the information in a form is directly accessible from the associated package.

Package
The basic unit of work that moves through a life cycle. It typically represents a problem or a request that needs to be tracked, the changes made in response to the problem or incident, and any other associated information.

Package Group
This provides a way to operate on related packages as a unit. A particular package might belong to one group or several, or it might not belong to any. Any operation that can be performed on a package can be performed on all packages in a package group.

Projects

This term refers to the entire control framework within Harvest that supports a particular development or maintenance activity. A project includes information about what data to access, how changes progress through the cycle, the activities that can occur, and user responsibilities. You may have many different projects, depending on the applications being controlled and the kind of development activity undertaken.

Life Cycles

A life cycle describes the path that changes take as development progresses in a project in terms of an ordered set of phases or states. The life cycle can be considered the "heart" of a project because it controls the flow of development life within it. A common development scenario might include states for assigning change requests, making changes, testing changes, integrating changes and releasing a completed project. A set of valid processes can be defined for each state in the life cycle. Processes are commands that perform a task. The processes defined for a state determine the activities that can be performed, or its scope of work. Additional actions can be associated with each process, based on the success of its execution. This allows commands to be linked together to perform more complex tasks.

User and User Groups

When Users are defined in Harvest, they can be assigned to User Groups. Users and User Groups exist at the global level within Harvest, so they are available to all projects defined within a Harvest installation. A User can belong to any number of User Groups, and there is no hierarchy implied by the groups. User Groups provide a powerful, flexible mechanism that can be employed in association with access control, notification and approvals.

Repositories

This is where data that is under Harvest's control, is stored. Repositories reside within the relational databases. Typically, a repository is comprised of all files that make up an application, but applications can also be split up between multiple repositories. Repository data can either be accessed directly by a user with Harvest Administrator access or other users can access it through views.

Views

Views define what repository data can be accessed from within a project. Each project has one baseline, which defines the repository items available to the project. A baseline can include items from more than one repository, as in the case of shared code. Additional views, called working views, can be created based on the baseline view. Within a life cycle, each state can have a particular view associated with it. The view determines which item versions are accessible to users operating in a state using that view. The third type of view, called a snapshot view, allows users to capture their application inventory at a specific point in time. This facilitates the re-creation of specific application versions.

Items

Repositories are composed of items. An item in the repository is equivalent to a file on the file system.

Versions

Each item in a repository consists of versions of the item. Item versions are visible to users only through views; they cannot be seen by looking directly at the repository. Base versions, which are the initial versions of the items in a repository, are shown through the baseline view and all working views. Subsequent versions created after the baseline view has been established, are shown through working views only.